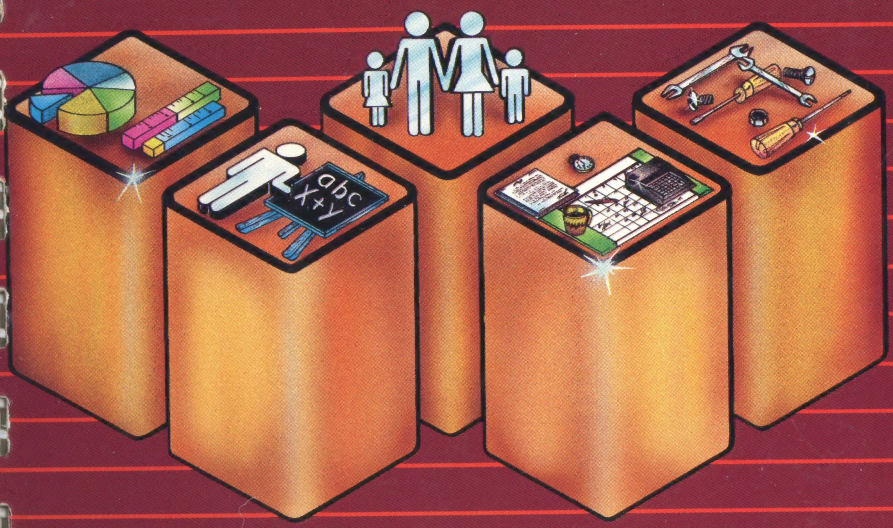
CHART 'N GRAPH TOOLBOX™

Graph Related Commands
For Applesoft

By Andy Scheck & Rick Chapman



Roger Wagner™
PUBLISHING, INC.

Chart 'n Graph Toolbox Addendum

• ProDOS Version
• Recent Changes to the Software

Booting the ProDOS version of the Chart 'n Graph Toolbox

There is no room on either side of the diskette for the PRODOS or BASIC.SYSTEM files. Therefore, to boot, do the following:

1) Boot on another ProDOS diskette, such as your ProDOS System Disk, and go to Applesoft BASIC.
2) Insert either side of the ProDOS Chart 'n Graph Toolbox, and type:
PREFIX,D1 (Return)
3) Type: RUN HELLO (Return)

Side One Notes - ProDOS version

* Most of the Toolbox command files have been renamed to conform to ProDOS naming conventions.
* The PRINTER.SETUP program has been modified to correctly address the renamed commands.
* BLOAD PIC.TB and BSAVE PIC.TB have been removed since they are not necessary in ProDOS.
* RELOCATE.I.TB and RELOCATE.II.TB have been modified for ProDOS. The DOS 3.3 and ProDOS versions of these commands are not interchangeable.
* The file called UNRELOCATE on the ProDOS Chart 'n Graph diskette does not return to the calling program. Instead, it does a warm start since ProDOS does not contain the FP command.
* The HELLO program uses RELOCATE.II.TB, and therefore AMPERCHART is no longer installed at the top of memory. This also means that a number of the demos that load other Toolbox commands have been modified to load those commands into different locations.

Side Two Notes - ProDOS version

* The following DHR commands have been modified: DHR/RELOCATE.TB, DHR/DISPLAY.TB, DHR/ROUTINES, and DHR/AMPERCHART.
* DHR LOAD.TB and DHR.SAVE.TB have been deleted. To save in a standard DHR format from BASIC, use the following:

```
&"TRANSFER",1,2,1
&"TRANSFER",1,1,0
POKE 8192+120,2
PRINT CHR$(4);"BSAVE DHR.PIC,A$2000,E$5FFB,T$08"
&"TRANSFER",2,1,0
POKE 8192+120,2
PRINT CHR$(4);"BSAVE DHR.PIC,A$2000,E$EFFB,B$2000,T$08"
```

* The file called UNRELOCATE on the ProDOS Chart 'n Graph diskette does not return to the calling program. Instead, it does a warm start since ProDOS does not contain the [illegible] command.
* DHR/DHR.AMPERCHART and DHR/DHR.ROUTINES require that the 80 column card is activated prior to use. If the card is not selected, all text will be displayed as garbage. Just issue a simple PR#3 to avoid this problem. These two files are also compatible in DOS 3.3 if CONVERTed.
* DHR/DHR.AMPERCHART has the following internal locations, relative to the result of &LOC(X) which is $800:

DRWSET	$9A9	RESET	$EC0
HLBROT	$A1A	SCLSTR	$9A3
OLDC	$12DF	TABLE	$12EF
OLDX	$12DB	VLBROT	$AC8
OLDY	$12DD	VERSION	$19

* DHR/RELOCATE.TB loads the files DHR/AMPERCHART and DHR/ROUTINES. Therefore, these files must be available under the prefix DHR/ when DHR/RELOCATE.TB is called.
* The relocate command uses a large portion of page 2, so the line calling it must not be longer than 80 characters. This should not be a problem since ProDOS commands must be the first command on a line. To avoid problems with this, do not use extra-long pathnames (more than 4 subdirectories deep), or do not execute it from the immediate mode at the end of a long input line.
* The HELLO program no longer includes the option to modify Applesoft.
* The shape module commands were modified. The configuration program will only run under ProDOS.
* The shape modules are compatible with both DOS and ProDOS versions of the Chart 'n Graph Toolbox.
* It is recommended that you print out a catalog listing of both sides of the diskette for your reference.

Error Alert! (ProDOS version only)

When using RELOCATE.I.TB or RELOCATE.II.TB under ProDOS, you may occasionally get a seemingly unexplicable "SYNTAX ERROR" on the line that calls the RELOCATE command. This is usually due to a bug internal to BASIC.SYSTEM and ProDOS. It is often brought on by using the ProDOS command "PREFIX/" to restore the prefix to the root level. The result is that BASIC.SYSTEM and ProDOS get "out of sync" with each other with respect to the current prefix. This can be avoided by changing the beginning of your ProDOS programs that use RELOCATE to look like this:

```
1  CALL PEEK(175) + 256 * PEEK(176) - 46
2  PRINT CHR$(4);"PREFIX": INPUT P$: PRINT CHR$(4);"PREFIX";P$
3  &"RELOCATE"
```

This will correctly re-synchronize BASIC.SYSTEM and ProDOS so that the RELOCATE command will function properly.

Other Changes...

NEW INTERNAL ENTRY POINTS AND DATA LOCATIONS

Recent changes have been made to the Amperchart Toolbox Files related to locating the internal entry points and data locations (see pages 169-171 for specific details).

All Amperchart Toolbox Files (AMPERCHART.PLOT.TB, AMPERCHART.PLOT+LABEL.TB, AMPERCHART.TB, DHR AMPERCHART.TB, and DHR AMPERCHART) now use the same system for locating internal entry points and data locations. The locations of the new entry points for each Amperchart Toolbox file, along with the Revision Number, are located just after the Version Number, at a fixed location relative to the beginning of the module where located in memory.

The following table specifies the locations of the entry addresses and data locations, relative to the value returned by &LOC(X):

Description	Relative Location
Version Number	$19
Revision Number	$1A
RESET	$1B-1C
OLDX	$1D-1E
HLBROT	$1F-20
VLBROT	$21-22
SCLSTR	$23-24
DRWSET	$25-26
TABLE	$27-28

Note that the final five addresses, HLBROT, VLBROT, SCLSTR, DRWSET and TABLE are not present for AMPERCHART.PLOT.TB since it does not support Hi-Res labeling and shape tables.

The locations for OLDY and OLDC are obtained by adding 2 and 4, respectively, to the resultant address for OLDX.

For example, to set HLBROT to 16, one could use the following BASIC statements:

```
10    &LOC(X): REM LOCATE CURRENT ADDRESS
20    AD = X + PEEK(X+31) + 256 * PEEK(X+32)
25    REM $1F,20 = 31,32
30    POKE AD, 16
```

2. IDENTIFICATION.TB

IDENTIFICATION.TB now gives the following results (Final value = R1 + R2).

R1	Comments
128	Has extended 80 column card
64	Has standard 80 column card
32	No Apple 80 column card
0	Not a IIe, IIc or IIGS

R2	Comments
0	IIe
1	IIe+ (Enhanced)
2	IIc
3	IIc (new Unidisk 3.5 ROMs)
4	IIGS
5	Unknown

You can use the following lines in your program to determine R1 and R2:

```
100  &"ID", R
110  T = INT(R/32): R1 = T * 32
120  R2 = R - R1
```

CHART 'N GRAPH TOOLBOX™

Graph Related Commands
For Applesoft

By Andy Scheck & Rick Chapman

INSTRUCTION MANUAL

Copyright © 1984 by Roger Wagner Publishing, Inc. All rights reserved. This document, or the software supplied with it, may not be reproduced in any form or by any means in whole or in part without prior written consent of the copyright owner.

ISBN 0-927796-20-1

PRODUCED BY:

Roger Wagner™
PUBLISHING, INC.

1050 Pioneer Way • Suite P • El Cajon, CA 92020
Customer Service & Technical Support: 619/442-0522

OUR GUARANTEE

This product carries the unconditional guarantee of satisfaction or your money back. Any product may be returned to place of purchase for complete refund or replacement within thirty (30) days of purchase if accompanied by the sales receipt or other proof of purchase.

Product Reference CHART 2C1087

*** THE CHART 'N GRAPH TOOLBOX ***

A library of new charting and graphics commands for Applesoft BASIC.

AMPERCHART Command Set by Andy Scheck
Additional commands by Rick Chapman

TABLE OF CONTENTS

1. INTRODUCTION

1.1 System Description 2
1.2 Diskette Contents 2
1.3 How To Use This Manual 4

2. THE TOOLBOX SYSTEM - INTRO SECTION

2.1 A Short Demonstration 5
2.2 If It Doesn't Work 8
2.3 Adding More Commands/Resuming Operation..... 9

3. AMPERCHART TUTORIAL

3.1 Getting Started 11
3.2 Adding AMPERCHART to a Program..... 12
3.3 Initialization & Mode Commands 14
3.4 Coordinate System 16
3.5 Move, Plot, Draw, and Clipping 17
3.6 Scaling and User Coordinates 18
3.7 Clipping Windows 20
3.8 Clears and Frame 22
3.9 Axis and Grid Generation 23
3.10 Labeling 26
3.11 Display and Working 29
3.12 Arc Draw 29
3.13 Area Filling 31
3.14 Logarithmic Mode 32
3.15 Screen-to-User Coordinate Conversion 32
3.16 Location Command 33

4. AMPERCHART COMMAND REFERENCE

4.1 GGO - Graphics Go 36
4.2 TMD - Text Mode 37
4.3 GMD - Graphics Mode 37
4.4 MIX - Mix Text and Graphics 38
4.5 NMX - No Mix 38
4.6 DSP - Set Displayed Page 39
4.7 WRK - Set Working Page 39

4.8 CLP - Clipping Window 40
4.9 SCL - Scaling 41
4.10 LMD - Log Mode 42
4.11 CLR - Clear Window 43
4.12 FCL - Clear Full Screen 43
4.13 RVS - Reverse Window 44
4.14 MOV - Move 44
4.15 PLT - Plot 45
4.16 DRW - Draw 45
4.17 BIT - Read Screen Bit 46
4.18 AXS - Axis 47
4.19 GID - Grid 48
4.20 HLB - Horizontal Label 49
4.21 VLB - Vertical Label 50
4.22 FIL - Fill Area 51
4.23 ARC - Draw Polygonal Arc 52
4.24 TSZ - Tic Size 53
4.25 FRM - Frame Window 53
4.26 STU - Screen-to-User Coordinate Conversion . 54
4.27 LOC - Location of Amperchart 54

5. OTHER LIBRARY FILES

5.1 Use With or Without the Workbench 55
5.2 RELOCATE I&II.TB/JUMP FILE..... 57
5.3 SPLITTER 59
5.4 BLOAD HI-RES IMAGE 63
5.5 BSAVE HI-RES IMAGE 64
5.6 WINDOWING 65
5.7 HI-RES/LO-RES ZOOM 68
5.8 HI-RES EXPAND..... 70
5.9 MEDIAN FILTER 71
5.10 THREE-D ARRAY TRANSFORMS 73
5.11 GRAPHICS DUMP COMMANDS 76
5.12 COMPUTER INDENTIFICATION 80
5.13 TONE 82
5.14 RESTORE AMPERSAND 84
5.15 Alternative Shape Tables 86
5.15.1 Using the New Shape Tables 86
5.15.2 Creating New Shape Tables 87
5.15.3 Using AMPERCHART Shape Tables 87
5.15.4 Shape Table Requirements 88
5.15.5 UC & LC SHAPE.TB 90
5.15.6 S10X14 SHAPE.TB 91
5.15.7 APPLE ICONS.TB 93

TITLE PAGE

6. SINGLE "RES" DEMO PROGRAMS - NOTES AND HINTS ... 94

7. SYSTEM USAGE AND INSTALLATION

7.1 Memory Usage in the Apple..... 100
7.2 Toolbox System: Standard Option 102
7.3 Using SPLITTER & AMPERCHART.TB 104
7.4 Stand-Alone Method: High Memory 105
7.5 Stand-Alone Method: Low Memory 107
7.6 Alternative Installation Methods 108

8. SPECIAL VERSIONS OF AMPERCHART.TB 109

9. DOUBLE HI-RESOLUTION PLOTTING

9.1 Double Hi-Resolution Mode 111
9.2 Hardware Requirements 112
9.3 Double Hi-Resolution Software 112
9.4 Double Hi-Res Command Sets 113
9.5 Double Hi-Res Commands 114
9.6 Double Hi-Res Command Options..... 115
9.7 Double Hi-Res Versions of AMPERCHART 116
9.8 Summary of Installation Options 117
9.9 DHR Dump Commands 118
9.10 Miscellaneous Support Commands 118
9.11 Double Hi-Res Demonstration Programs 118
9.12 DHR RELOCATE.TB/JUMP.TB 120
9.13 DHR AMPERCHART 121
9.14 DHR ROUTINES 122
9.15 DHR DISPLAY.TB 124
9.16 DHR LOAD.TB 125
9.17 DHR SAVE.TB 126
9.18 DHR WINDOW.TB 127
9.19 DHR TRANSFER.TB 130
9.20 DHR ROUTINES.TB 132
9.21 DHR PRINTER DUMPS 134
9.22 DHR RELOCATE.MOD.TB 137
9.23 DHR MODIFIER.TB 138
9.24 DHR AMPERCHART.MOD 143
9.25 DHR AMPERCHART.MOD.TB 145

10. THE WORKBENCH IN DETAIL

10.1 Workbench Main Menu 146
10.2 Adding Commands 147
10.3 Removing Commands 147
10.4 Copying All Commands to Disk 148

TITLE PAGE

10.5 Restoring Commands 149
10.6 Report Commands Added 150
10.7 Search for Toolbox Commands 151
10.8 The Memory Map 153

APPENDIX A - A Little More Detail

How the Workbench Uses Memory 157

APPENDIX B - Advanced Use of the Toolbox System

Using CALL Statements 161
Using the Ampersand 162
Restoring the Ampersand Vector 164
Using Jump File Create 165

APPENDIX C - Advanced Graphics Topics

Amperchart Version Number 168
Amperchart Internals 169
Graphics Flashing Cursor 172
Clearing to a Color 172
Multiple Screens - Super Hi-Res 172
Double Hi-Res Command Set Entry Points 173

APPENDIX D - EASY CHART

Running EASY CHART 174
Entering and Editing Data 174
Plotting the Data 175
Plotting Parameters
Labeling
Area Erase
Program Modification 178
DIF to Easy Chart File Converter 179

APPENDIX E - SHAPE MAKER PROGRAM

Introduction 181
Using Shape Maker 182
Joystick Files 184
Programming Notes 185

QUICK REFERENCE PAGES 186

ABOUT THE AUTHORS

Andy Scheck received his BS degree from Valparaiso University in Indiana and, after securing a position in Maryland, he obtained his MS in Computer Science from Johns Hopkins. Andy has been involved with computers for over a decade, and & CHART was modeled after the graphics instruction set of the HP9845A which he had access to at work. He is a fluent programmer in BASIC, FORTRAN, and Pascal, as well as 6502, 6800, 6809 and 68000 assembly languages. Recently married, Andy likes to swim, ski, golf and hang-glide, and his future plans include getting his priorities straight, and getting his Hang-2 rating.

Rick Chapman took a class in assembly language as an Eagle Scout and has been programming ever since. He has a BS in Physics from the University of Maryland, a MSEE from Johns Hopkins, and is currently working on his Doctorate in Oceanography at Florida State. Rick was writing Real Estate programs on his Apple when he met Andy and the result of their friendship was & CHART. Though Rick programs in BASIC, Pascal, C, and APL, he recently began studying 68000 assembly language in preparation for a Macintosh graphics package he is developing with Andy. Rick's goals are to continue playing lacrosse for FSU, to become an Astronaut for NASA, and to learn to dance like Michael Jackson.

First, our legal stuff...

ROGER WAGNER PUBLISHING, INC.
CUSTOMER LICENSE AGREEMENT

IMPORTANT: The Roger Wagner Publishing, Inc. software product that you have just received from Roger Wagner Publishing, Inc., or one of its authorized dealers, is provided to you subject to the terms and conditions of this Software Customer License Agreement. Should you decide that you cannot accept these terms and conditions, then you must return your product with all documentation and this License marked "REFUSED" within the 30 day examination period following the receipt of the product.

1. License. Roger Wagner Publishing, Inc. hereby grants you upon your receipt of this product, a nonexclusive license to use the enclosed Roger Wagner Publishing, Inc. product subject to the terms and restrictions set forth in this License Agreement.

2. Copyright. This software product, and its documentation, is copyrighted by Roger Wagner Publishing, Inc. You may not copy or otherwise reproduce the product or any part of it except as expressly permitted in this License.

3. Restrictions on Use and Transfer. The original and any backup copies of this product are intended for your personal use in connection with a single computer. You may not distribute copies of, or any part of, this product without the express written permission of Roger Wagner Publishing, Inc.

P.S. We have tried our best to give you a quality product at a fair price, and made the software copyable for your personal convenience. Please recommend our product to your friends, but respect our wishes to not make copies for others. Thanks!

LIMITATION ON WARRANTIES AND LIABILITY

ROGER WAGNER PUBLISHING, INC. AND THE PROGRAM AUTHOR SHALL HAVE NO LIABILITY OR RESPONSIBILITY TO PURCHASER OR ANY OTHER PERSON OR ENTITY WITH RESPECT TO ANY LIABILITY, LOSS OR DAMAGE CAUSED OR ALLEGED TO BE CAUSED DIRECTLY OR INDIRECTLY BY THIS SOFTWARE, INCLUDING, BUT NOT LIMITED TO ANY INTERRUPTION OF SERVICE, LOSS OF BUSINESS OR ANTICIPATORY PROFITS OR CONSEQUENTIAL DAMAGES RESULTING FROM THE USE OR OPERATION OF THIS SOFTWARE. SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF IMPLIED WARRANTIES OR LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY TO YOU.

Then Apple's...

DOS 3.3 Standard is a copyrighted program of Apple Computer, Inc. licensed to Roger Wagner Publishing, Inc. to distribute for use only in combination with The Chart 'n Graph Toolbox.

APPLE COMPUTER, INC. MAKES NO WARRANTIES, EITHER EXPRESS OR IMPLIED, REGARDING THE ENCLOSED SOFTWARE PACKAGE, ITS MERCHANTABILITY OR ITS FITNESS FOR ANY PARTICULAR PURPOSE. THE EXCLUSION OF IMPLIED WARRANTIES IS NOT PERMITTED BY SOME STATES. THE ABOVE EXCLUSION MAY NOT APPLY TO YOU. THIS WARRANTY PROVIDES YOU WITH SPECIFIC LEGAL RIGHTS. THERE MAY BE OTHER RIGHTS THAT YOU MAY HAVE WHICH VARY FROM STATE TO STATE.

And from DSR...

This disk contains a high speed-operating system called Diversi-DOS(tm), which is licensed for use with this program only. To legally use Diversi-DOS with other programs, you may send $30 directly to: DSR, Inc., 5848 Crampton Ct., Rockford, IL 61111. You will receive a Diversi-DOS utility disk with documentation.

Diversi-DOS(tm): Copyright 1982 DSR, Inc.

And now on with our program!

INTRODUCTION

WELCOME TO THE TOOLBOX SERIES!

The Toolbox Series is the beginning programmer's dream, but it also appeals to intermediate as well as advanced Applesoft programers.

The Toolbox Series is a library of new commands for Applesoft that make writing ANY program in Applesoft quick and simple. It is viewed by many people as the "Applesoft Construction Set" because it allows you to build the programs you want out of an extensive set of "program building blocks" called Toolbox Files.

The Chart 'n Graph Toolbox makes putting graphics and charts in your programs quick and easy. You won't need to find, research, write or debug these essential parts of your programs. With the Chart 'n Graph Toolbox, if you wanted to draw a frame around the graphics screen, all you need to do is add the Toolbox command to draw a frame! The new frame command would look something like this:

```
100 & "CHART", FRM
```

and instantly a frame would appear on the screen around any Hi-Res picture you were creating.

How does it work? That's the nice thing about the Toolbox system; all you need to do is decide what new command you'd like to have, and then use the Workbench utility to add the new command to your program.

The Toolbox Series has many libraries in it, each of which add from 25 to 40 different commands to ordinary Applesoft BASIC.

The Chart 'n Graph Toolbox package has one large command set that adds 27 new graphics commands all at once to Applesoft, and other individual commands that can be added one at a time when you need them. And, if you have an Apple //e with 128k or an Apple //c, the Chart 'n Graph Toolbox will give you all the new commands you need to use the DOUBLE HI-RES GRAPHICS capabilities of these machines for even more detailed graphics and charts!

The Database Toolbox has over 40 array commands, including searching and sorting and different kinds of array manipulations; The Video Toolbox has over 30

text screen input and output commands; The Wizard's Toolbox has over 30 of the most often needed commands including PRINT USING, SOUND EFFECTS, and HI-RES TEXT commands. Try one, and you'll NEVER program without the Toolbox again!

1.1 About the Chart 'n Graph Toolbox

The Chart 'n Graph Toolbox is composed of two main parts.

The first is AMPERCHART itself, which is a "pre-grouped" library of 27 'EXTRA' graphic oriented Applesoft commands. They can be used on any Apple II family computer. Although each of the commands performs a single function, these commands were grouped into one unit because they are almost always used in one total combination in any given program.

Also included on the Chart 'n Graph Toolbox diskette is a variety of additional individual graphics support commands. These include commands to support the new Double Hi-Resolution graphics mode on the Apple //c, or an Apple //e with an extended memory 80 column card. These commands allow you to easily use Double Hi-Res graphics from within your own program, either with or without AMPERCHART. In addition, there are graphics dump commands, windowing commands, and a splitter program designed to split or unsplit Applesoft programs around the Hi-Res pages.

All Toolbox commands are added to your program using a utility called the Workbench. The Workbench lets you add just the command you're interested in at the time, and also provides "librarian" functions that let you list all added commands, delete unwanted commands, etc.

1.2 Chart 'n Graph Toolbox Diskette Contents

The Chart 'n Graph Toolbox diskette is double sided. The front side is oriented to the normal Hi-Res graphics mode on the Apple. The second side contains the double Hi-Res version of AMPERCHART and other double Hi-Res commands. There is also the EASY CHART data plotting program, and the SHAPE MAKER utility.

You do not have to boot on the Chart 'n Graph Toolbox diskette to use any of the programs on the diskette. If the power was not previously on, or if you wish to install the modified disk operating system (DOS)

used on the Chart 'n Graph Toolbox diskette, you may, however, boot in the usual manner. This will not effect any programs you may wish to run later.

IMPORTANT: It is strongly recommended that you do not use your originally purchased diskette, in either the examples to follow, or your daily use of the Chart 'n Graph Toolbox diskette. Instead, make a back-up using any normal diskette copy program, such as Apple Computer's COPYA, or COPY-CAT, produced by Roger Wagner Publishing. The original should then be put in a safe place, and the back-up used as your work diskette. After making your work diskette, return to this section.

When the front side of the Chart 'n Graph Toolbox disk is first booted, the HELLO program will give you the option of running any of the demos on the disk, the disk based AMPERCHART tutorial, the Workbench, or just installing AMPERCHART and proceeding directly to Applesoft BASIC.

The HELLO program menus on each side of the Chart 'n Graph Toolbox diskette allow you to use either the arrow keys, or the direct entry of a number for your choice. After selecting an item, press the Return key to complete the selection. Whenever a demo program is done, you can press any key to return to the main menu. Pressing ESCAPE in the main menu program will always back you up to a previous level, as indicated by the prompt in the upper right corner of the screen.

Booting the back of the diskette runs a menu program allowing you to try the Double Hi-Res routines, run the data plotting program EASY CHART, modify Applesoft if you have an Apple IIE, or run other utility programs included on the diskette.

SPECIAL DOS FEATURES

If you do boot on the Chart 'n Graph Toolbox diskette, a special Disk Operating System (DOS) will be in effect which you may find to be of some advantage.

The first thing you will notice during a CATALOG is a number just to the right of the 'RWP VOLUME 254'. This is the number of free sectors remaining on the diskette being CATALOGed. A normal Apple DOS 3.3 diskette has a maximum of 496 free sectors available.

The second feature is an optional termination of the CATALOG listing at any of the pauses which usually occur when a directory has more files than can be displayed on the screen at one time. When using the modified DOS, pressing RETURN will terminate the CATALOG listing at any pause. Pressing any other key will continue it.

1.3 How to Use This Manual

This manual consists of a tutorial on the use of the Workbench utility itself, a description of AMPERCHART system usage and syntax, a description of the additional Toolbox commands that are included in this package (including the Double Hi-Res commands), and an AMPERCHART command reference section.

The beginning sections of the manual, describing AMPERCHART and its use, use only standard Hi-Res graphics. The descriptions of the Double Hi-Res support commands are grouped together at the back of the manual for those users that have the required hardware (Apple //c or Apple //e with EXTENDED 80 column card).

Chapter 6 lists the standard AMPERCHART demonstration programs that are included on the AMPERCHART Diskette. The demos are pretty much self explanatory, and are provided to demonstrate actual applications of AMPERHCHART and the various additional commands.

EASY CHART, a menu-driven plotting program is described in Appendix D. It is included to give you a simple to use plotting program while at the same time providing a demonstration of all of the power of AMPERCHART.

The best way to use the manual is to first read Chapters 2 and 3 on the Workbench and the AMPERCHART command set. Then go through the table of contents and look for the commands that interest you, and read the sections of the manual devoted to just those commands.

After you have used the Chart 'n Graph Toolbox for a while, you will also find that the Command Reference. (Chapter 4) and the Advanced Topics section (Appendix C) are useful sections to be aware of. There are also quick reference pages located in the rear of the manual in order to make it easy to find the syntax for a given command as you need it.

2.1 A SHORT DEMONSTRATION

The primary purpose of the Chart 'n Graph Toolbox is to add new commands to your own Applesoft programs, as easily and efficiently as possible.

The Toolbox uses a utility called the Workbench to actually add the new commands to your programs. The Workbench is located on side 1 of your diskette. To use it, all that you have to do is:

1) Insert one special line in your program that tells Applesoft you have added some new commands to your program.

2) Use the Workbench utility to add the commands of your choice.

3) Use these commands from within your Applesoft program by using the ampersand followed by the name of the new command.

To see how this works, let's consider an actual example. Let's suppose for a moment that you would like to have some kind of musical sounds in a program you are starting to write. All you need to do is:

1. Type in FP to clear any program in memory. (If your program was already in memory, you wouldn't do this step.)

2. With the Toolbox disk in the drive, type in

   EXEC AMPERSAND SETUP. Then type in LIST and you should see:

   1 CALL PEEK(175) + 256 * PEEK(176) - 46

When your program is run this line will connect Applesoft to the various Toolbox commands you have added. (You could type this line in yourself anytime you want, but the EXEC method makes things easier.)

3. Get the Workbench installed by typing in:

   BRUN WORKBENCH

After a few seconds the main menu will appear:

```
          ***  TOOLBOX WORKBENCH ***

        SELECT AN OPTION:

     1.  ADD A COMMAND
     2.  REMOVE A COMMAND
     3.  REMOVE ALL COMMANDS
     4.  COPY ALL ADDED COMMANDS TO DISK
     5.  RESTORE COMMANDS FROM DISK
     6.  REPORT COMMANDS ADDED
     7.  SEARCH FOR TOOLBOX COMMANDS
     8.  DISPLAY MEMORY MAP
     0.  EXIT
                                [NO COMMANDS ADDED]
```

4. Since the first thing we want to do is to add a new command to our program, press '1'. The screen will then display:

```
INSERT DISK WITH TOOLBOX FILE TO BE
ADDED, THEN ENTER NAME OF FILE
('CAT' FOR CATALOG, <RETURN> TO QUIT)
TOOLBOX FILENAME ->
```

If you knew that TONE.TB was the file for the command you wanted to add at this point, you could just enter the name directly. Often you will not know the name exactly, in which case you can enter 'CAT' to produce a catalog of the diskette. Enter 'CAT' now. Since TONE.TB is further down the list, press the space bar (not RETURN) for the next page of the catalog, and continue until you can see TONE.TB. Now press RETURN to stop the catalog (assuming that you have booted with the Wizard's Toolbox diskette).

5. Now type in TONE.TB as the Toolbox File to be loaded. Then press Return. Note that all Toolbox File names have a '.TB' suffix.

Next the screen will display:

```
YOUR COMMAND NAME ->
```

6. You must now enter the new name which you will use for this command within your program. Because the Toolbox Files themselves may have rather long names (so as to be most descriptive) you will often want to use a shorter name when using the routine from within your program. In this case, 'TONE' will do

nicely. Type in TONE as the name and press RETURN.

The Workbench will now load the TONE.TB routine, and add the command 'TONE' to your program.

After the command is added, the program will return to the request for a new FILENAME. This is because you may want to add several commands at one time. Press RETURN alone to return to the menu at this point.

7. Now enter '0' and press RETURN to exit the Workbench and return to your program.

8. When using these new commands in your Applesoft programs, a special 'syntax', or command structure, is required. Each Toolbox command has either two or three parts, as follows:

The first part required is the ampersand symbol (&). You can think of this as being similar to the Control-D character needed whenever a disk command is done. The difference here is that the ampersand will call a Toolbox command.

The second item required is the command name (in quotes) for the routine you wish to call. For our example, this would be "TONE". If the command which you have just added does not require any information to be passed to it, then nothing more is required in the command.

The third item is optional, and is referred to as a "variable list". Each Toolbox command may or may not have certain pieces of information which have to be passed to it. In the case of the TONE.TB routine, this information is pitch and duration. (In the case of other commands, the variable list following the name may have a different number of variables required and use a variety of variable types.)

To use your new "TONE" command in a program, you would first determine the values for pitch (P) and duration (D), such as through an INPUT statement, and then call the command via the ampersand statement. To show how this might be done, enter the following program lines (these will now be in addition to the line #1 which was previously entered via the AMPERSAND SETUP procedure above):

```
10 INPUT "ENTER PITCH, DURATION: "; P, D
```

```
20 & "TONE", P, D
30 PRINT: GOTO 10
```

9. Now just RUN the program to try it out! Enter:

```
10,10
50,10
100,20
```

as some sample number pairs. It's that easy!

By the way, remember that, like normal Applesoft, you can use any variable names you want in calling a command. As long as you include the type of variables that the command expects, the variable name (or expression) is up to you.

YOU'RE DONE! You now have a complete program. The program can be LOADed or SAVEd just like any normal Applesoft program, and the new commands will be carried along automatically. They will remain a permanent part of your Applesoft program, unless you decide later to remove them using the Workbench "Remove a Command" option.

If you want to add more commands later, just BRUN WORKBENCH again, and add the commands you want. See the section later in this manual to determine the exact syntax for using each command.

2.2 IF IT DOESN'T WORK

In order for your program to use an added command it must meet two conditions: The ampersand must have been set up before any Toolbox command is executed in your program, and there must actually be an added Toolbox File with the name used in your command. If either of these conditions is not met, your program will probably crash or hang.

THE AMPERSAND LINK. It is good practice, when developing a program which will use Toolbox commands, to BEGIN by EXECing the AMPERSAND SETUP file. As explained earlier, this inserts a line (with line #1) in your program which tells Applesoft that you are using the Toolbox commands.

This line can occur anywhere in your program, provided that it is executed prior to the first Toolbox command. If it is not present then you'll probably get

a SYNTAX ERROR when you try to use one of your new commands. If this happens when you run your program, check to make sure the ampersand setup line is present AND being executed prior to the Toolbox command.

THE TOOLBOX FILE. Another way to bomb your program is to attempt to use a Toolbox command which has not been added. In this case (assuming that the proper setup line has been inserted) your program will try to find the new command named by you among the commands added. When it doesn't find it, you will get an UNDEFINED FUNCTION error message. You can then determine which Toolbox command is missing.

If you are sure that both the ampersand link and added command are present, then double-check the syntax (and perhaps the sample program listing) for the command as listed in the section on Toolbox Library Files in this manual.

2.3 ADDING MORE COMMANDS / RESUMING NORMAL OPERATION

If you want to add a new command right away, type in CALL 2051 to re-enter the Workbench. If you're done using the Workbench for a while and want to free up the memory normally occupied by the Workbench utility, type in: BRUN REMOVE WORKBENCH. This will remove the Workbench from memory.

The Workbench utility is required in memory only when you are adding or removing commands (or doing any of the other things that it allows you to do). It is not required to be in memory when you run your program.

After you have added a command with the Workbench, you may exit and immediately run your program (with the Workbench still present).

If you are using Hi-Res graphics you should remove the WORKBENCH. To remove it BRUN REMOVE WORKBENCH as described above and then run your program as usual. IMPORTANT: If you are using Hi-Res graphics, be sure to read the section on the MEMORY MAP option of the Workbench!

If you need to know more about the memory usage by the Toolbox and Workbench see "A Little More Detail" in Appendix A of this manual.

SPECIAL TIP: FAST RUN METHOD

If you have booted on the Toolbox diskette, then you can take advantage of its special DOS to make using the Workbench even easier.

This is done by using the "wild card" option built into the DOS. When typing the name of a file, you can use just the first few letters of the name followed by an equal sign ("="). For example, if you wanted to BRUN the Workbench, you could just type:

BRUN WORK=

This tells DOS to BRUN the first file in the catalog starting with the letters 'WORK'. The "=" is called a wild card because it can represent any combination of remaining letters in the name.

Because the Workbench, Ampersand Setup, and Remove Workbench are the first files on the diskette catalog, it gets even easier to use them:

To install the Workbench, just type: BRUN W=

To add the Ampersand Setup line, type: EXEC A=

To remove the Workbench, type: BRUN R=

DEMONSTRATION PROGRAMS

There are a number of demonstration programs on both sides of the Chart 'n Graph Toolbox package. These were created using the Workbench to add commands from the Toolbox Files on the disk. These programs may be LISTed and studied to see exactly how the added commands are used within an Applesoft program.

IMPORTANT NOTE: The Workbench utility may be used to append ANY ampersand-called machine language routine to an Applesoft program, and is not limited to the Toolbox Files available on this or other Toolbox disks. This means it is possible to use routines that you have written yourself, or are listed in magazines. The only requirement is that the routine be location independent (i.e. it can't have JMP's and JSR's to labels within the routine itself).

3. AMPERCHART TUTORIAL

This chapter provides a tutorial on how to use AMPERCHART and what some of its more often used commands are. More information on specific commands and additional commands on the Chart 'n Graph Toolbox disk are found later in this manual.

3.1 Getting Started

AMPERCHART is an extraordinarily powerful graphics plotting and charting package designed to enhance the graphics capabilities of the Apple II family of computers. It is composed of 27 commands that will greatly simplify the task of using the Apple's Hi-Res graphics for the presentation of your data from within your Applesoft program. These 27 commands are like a library of graphics utilities that act as an extension to the existing set of Applesoft BASIC commands and functions.

These commands will, for example, allow you to select which portion or "window" of the Apple screen you will be plotting your data. They will allow you to redefine the Apple coordinate system so that you are no longer required to adjust your data from 0-279 and 0-191. You could decide, for example, that the left side of your selected window is -1000, the right side is +10, the bottom is 0 and the top is 500. Then when you plot your data, if it's within your window's limits, you'll see it. If it's outside the window, it will be automatically "clipped", thus avoiding the usual Applesoft ILLEGAL QUANTITY ERROR.

Commands are available for the generation of axes with tic marks and grid lines, and of course you have the ability to produce alphanumeric labels with a variety of positioning options.

This tutorial is a collection of examples and techniques that should give you a feeling for the power that AMPERCHART affords the user in the presentation of graphical data in various formats. Each of the 27 commands will be presented in varying depth and detail.

See the command reference chapter for a complete description of the actions invoked by each command. As examples are presented, you'll find it instructive to have your Apple follow along.

3.2 Adding Amperchart to a program

The first step is to create a program that will serve as the framework within which we will explore the abilities of AMPERCHART.

To do this we will need to follow the three basic steps described in Chapter 2:

1) Insert one special line in your program that tells Applesoft you have added some new commands to your program.

2) Use the Workbench to add the commands to your program.

3) Use these new commands from within your Applesoft program by using the ampersand followed by the name of the added command.

To start, first boot on side 1 of the Chart 'n Graph Toolbox diskette, and then press ESCAPE to exit the main menu to Applesoft BASIC. Then:

1. Type in 'FP' to clear any program in memory.

2. With the Chart 'n Graph Toolbox disk (side 1) in the drive, type in EXEC AMPERSAND SETUP. Then type in LIST and you should see:

   ```
   1 CALL PEEK(175) + 256 * PEEK(176) - 46
   ```

   When your program is run, this line will establish a connection from Applesoft to the AMPERCHART commands you are about add.

3. Get the Workbench up and running by typing in:

   ```
   BRUN WORKBENCH
   ```

4. There are two commands which have to be added to use AMPERCHART. They are called RELOCATE I.TB and AMPERCHART JUMP FILE.TB.

   Because of some memory usage items that will be discussed in more detail later, AMPERCHART.TB is best used by being loaded into memory at a location different from that of your Applesoft program.

This is somewhat different than virtually every other Toolbox command you will use. This is because AMPERCHART is rather large (being a collection of 27 commands), and is best utilized by finding a spot in memory that is just the right size for it.

The Toolbox file RELOCATE I.TB does the loading operation, while AMPERCHART JUMP FILE.TB tells your program where AMPERCHART.TB is in the computer.

Thus, the only two commands you need to add to use AMPERCHART.TB are RELOCATE I.TB and AMPERCHART JUMP FILE.TB. To do this, select menu item #1.

5. Although you could CATALOG the disk if you were unsure of the spelling of the command to add, in this case you can just type in:

   RELOCATE I.TB

   as the Toolbox file to be loaded.

   The screen will then display:

   NEW COMMAND NAME ->

6. For this command, the name RELOCATE will be just fine. Type this in and press RETURN.

7. After the command is added, the program will return to the request for a new TOOLBOX FILE. At this point enter the name:

   AMPERCHART JUMP FILE.TB

   and press RETURN. When this is loaded, give it the command name:

   CHART

   This will be the command name that is used whenever an AMPERCHART command is needed.

8. Now enter '0' and press RETURN to exit the Workbench and return to your program.

9. Finally, type in:

   BRUN REMOVE WORKBENCH

This will remove the Workbench from the computer to give us room to use the Hi-Res display. THE WORKBENCH MUST ALWAYS BE REMOVED BEFORE RUNNING ANY PROGRAM THAT USES THE HI-RES SCREENS!

3.3 Initialization and Mode Commands

Now that the proper commands have been added to our program, we need to write the BASIC lines to use them. If you were to LIST your program now, it would just look like this:

```
1 CALL PEEK(175) + 256 * PEEK(176) - 46
```

This line activates the ampersand (&) character so the new commands can be used. Although you can't see anything in the listing, the commands that the Workbench added are actually there!

To use them, the second line in your program should be:

```
2  &"RELOCATE"
```

This will call the command that loads AMPERCHART.TB from the diskette and also sets up memory for best use of the Hi-Res pages. This line should always be done right after line 1 has set up the ampersand because if used after other variables have been defined in your program (like X=5 or D$=CHR$(4)) those variables will be lost. This is essentially the same restriction that applies to the use of HIMEM: or LOMEM: from within an Applesoft program. IMPORTANT! AMPERCHART.TB must be on the diskette in the drive when your program is RUN, so that RELOCATE can properly load it.

Now to actually use AMPERCHART. This is done by using the "CHART" command combined with other specific commands. Let's see how this is done.

The first item of business is to initialize some internal variables used by AMPERCHART. This is accomplished with the GCO (Graphics GO) command. With this command you must also specify which Hi-Res page you want to start with. This is very similar in function, but not identical, to the Applesoft HGR and HGR2 commands. Let's start by initializing page 1:

```
10  &"CHART",GCO(1)
```

GGO initializes AMPERCHART and displays page 1 of Hi-Res graphics. This usually is done only once, somewhere near the beginning of your program. You'll notice that AMPERCHART commands are all preceded by the general name "CHART" that we gave to the AMPERCHART JUMP FILE.TB command. This is the format that is used to execute an AMPERCHART command when it is being used as part of your program using the Toolbox system. Type in RUN now and see what happens.

The disk should come on for a second (during which time AMPERCHART.TB is loaded into the computer), and then the screen should clear to black as the High Resolution graphics screen is initialized.

If you type something now, you'll notice you can't see what you are typing. This is because the graphics display (as opposed to the text display) is active.

To return to the text screen, press RESET, and then add this line to your program:

```
20 &"CHART",TMD
```

Now type RUN again and see what happens.

The first thing to notice is that the disk did not come on this time. This is because RELOCATE can tell that AMPERCHART.TB was already loaded, and thus did not repeat the action.

This time, the program went to the graphics mode, and then immediately went back to the text mode. TMD stands for 'Text MoDe', and is used to switch the display back to text.

Although TMD is similar to the TEXT command in Applesoft, TMD will not reset the scrolling window parameters of the text window, just in case you modified them for some fancy text presentation.

The GMD (Graphics MoDe) does just the opposite of TMD. With these two commands you can switch back and forth between graphics and text mode. Note that neither command will affect the contents of the Hi-Res or text page. When the GMD command has been selected, the MIX (MIX mode) command can be used to display the mixed text and graphics with 4 lines of text at the bottom.

The other option, NMX (No MIX), will display the full graphics screen.

To see how this works, without using line numbers, or running your program, just type in:

```
&"CHART",GMD
```

The screen should immediately go back to the graphics display. You'll notice here that AMPERCHART can also be used in the immediate mode!

At this point, you're back to not being able to see what you're typing, so let's try another AMPERCHART command, MIX, which will put the display in a mode which displays both text and graphics.

To try this, type (I know it'll take a little concentration!) in:

```
&"CHART",MIX
```

If you don't see the cursor after you type this, press RETURN a number of times to bring the cursor down to the bottom of the screen.

You should now see a blank graphics screen and some text at the bottom. What we've covered so far are some of the basic graphics/text control statements that determine what's to be viewed. Since we haven't yet covered any commands that actually place anything on the screen, type the following:

```
&"CHART",RVS
```

This is the RVS (ReVerSe) command. Its function is to complement the color of the current screen. Since you started with a black screen, the result is a white screen. If you try it again, the screen will return to black again. At first this might seem to be a somewhat frivolous command, but later when you have some plotted data on the screen, you'll find it very useful for highlighting.

3.4 Coordinate Systems

Before we introduce any more commands, it's probably worthwhile to briefly review the way the Apple handles the coordinate system for its Hi-Res screen, because AMPERCHART handles it a little differently. The X-axis has a resolution of 280 points numbered 0-279 from left to right. The Y-axis has a resolution of 192 points numbered 0-191, and, unfortunately, they are numbered from top to bottom.

This puts the origin (0,0), somewhat awkwardly, at the top left of the Hi-Res screen. In AMPERCHART, the deviation from Apple's convention is that the Y-axis has been renumbered, still 0-191, but it's numbered from bottom to top. This puts the origin (0,0) at the bottom left...where it should be.

3.5 Move, Plot, Draw and Clipping

AMPERCHART contains many different commands, some of which are used more often than others. We will now introduce the real workhorses of AMPERCHART: PLT (PLoT), DRW (DRaW), and MOV (MOVe). Those commands actually put the dots and lines on the screen. Each of these three commands requires that you supply an X,Y coordinate pair. PLT (PLoT) will place a single dot (in the current HCOLOR) at your specified X,Y coordinate. DRW (DRaW) will draw a line (in the current HCOLOR) to the specified coordinate from the last PLT'd, DRW'n, or MOV'd point.

In the case of the MOV command, nothing is actually drawn on the screen, but an invisible cursor is placed at the selected X,Y coordinate. This command is usually used to establish a starting point for a subsequent DRW command. OK, I can hear it now: "Big deal, you can do all that with Applesoft's HPLOT". You're absolutely right, so for our example we'll do something HPLOT can't do. Do you remember what happens when you try to specify a coordinate outside the normal Hi-Res screen, i.e. either < 0 or >191 (>279 for the X axis). Go ahead and try it. You'll find that Applesoft is very adamant about the range of coordinates it will accept. When you're done, re-write, then RUN the sample program to try the following:

```
 1 CALL PEEK(175) + 256 * PEEK(176) - 46
 2 &"RELOCATE"
10 &"CHART",GCO(1)
20 &"CHART",MOV(10,-400)
30 &"CHART",DRW(140,800)
40 &"CHART",DRW(270,-400)
```

What did we do? The first graphics command, GCO, initialized everything just like before. The second command, MOV(10,-400), moved the invisible plotting cursor to somewhere near the floor, under the left-hand side of your viewing screen. The third command, DRW(140,800), drew a line from the floor to a point near the ceiling, directly above the center of your screen.

The final command, DRW(270,-400), drew another line from the ceiling to the floor, beneath the right side of your screen. Obviously, you are going to see only those portions of the lines that actually overlap your viewing screen. Thus in a very real sense, AMPERCHART allows you to go beyond the spatial confines of your screen, and plot to an area many times larger than your screen.

Actually your screen should be considered a window through which you can view your work on this much larger, somewhat imaginary, plotting surface. That's in itself quite powerful, and not easily accomplished in Applesoft. And now as we go on, we'll really start to extend this plotting capability.

3.6 Scaling and User Coordinates

Have you ever tried, without the advantages of AMPERCHART, to plot some data to the screen? If your data was in an array, you probably had to search the array for the largest and smallest values, then you calculated some conversion factor so the converted data would range from 0-191 (Y) and 0-279 (X), thus staying within the limits of the Hi-Res screen. Or suppose you thought you knew what the scale factor would be, and along comes a data point which was a bit bigger or more negative then you anticipated... (BEEP) ILLEGAL QUANTITY ERROR. Look familiar?! Well no more! AMPERCHART completely eliminates these annoying scaling problems.

You no longer have to bother about adjusting your data so it ranges from 0-191 or 0-279, because AMPERCHART allows you to determine the range of values to be displayed on the Hi-Res screen. This is done with the SCL (SCaLe) command, with which you simply specify the values that the edges of your viewing window are to assume, and then proceed to PLT, DRW, and/or MOV your data.

Remember that what you are doing with the SCL command is actually redefining the coordinate system of the Apple screen (which has a maximum range of 0-191 and 0-279 without AMPERCHART), to your coordinate system, which has whatever range you want. In order to make it easier to distinguish between X,Y coordinates in the two systems, throughout the rest of this manual the Apple's screen coordinates are called screen coordinates, and your coordinates are called user coordinates...clever!?

As an example, suppose you wanted to plot the function SIN(X) from X= -3pi to +3pi. The Y values of this function range from +1 to -1. The procedure for setting up the SCL factor can be demonstrated by entering and running the following program.

It's possible to do this in the immediate mode, but it'll be much easier to make changes if the program is entered in the deferred mode. Don't bother with the REM statements, they're just there to document our intent. Also be sure to delete any lines from the previous program except lines 1 and 2.

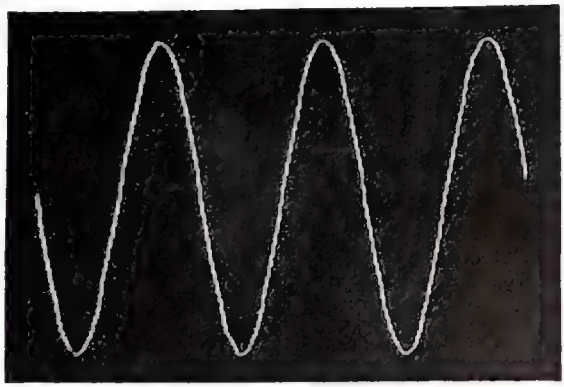
```
  1 CALL PEEK(175) + 256 * PEEK(176) - 46
  2 &"RELOCATE"
 10 ONERR GOTO 400 :REM Just in case of a mistake
100 &"CHART",CGO(1): REM Reinitalize everything, so
    we all start at the same place.
140 PI = 3.1415927:  REM Initialize the variable
    PI, to save time later
160 &"CHART",SCL(-3*PI,+3*PI,-1,+1): REM See below.
```

What's been done in line 160 with the SCL command is to define the left-hand side of the viewing window to be -3pi (-9.42478), the right-hand side is +3pi (+9.42478), the bottom edge is -1 and the top edge is +1. It's that simple to redefine the screen's coordinate system. Now to generate the data and plot it.

```
180 FOR X = -3*PI TO 3*PI STEP .15: REM
    FOR/NEXT Loop to generate data
200 &"CHART",PLT(X,SIN(X)): REM AMPERCHART do your
    stuff
220 NEXT X
240 VTAB 23: REM Send cursor to bottom of Text page
260 &"CHART",MIX: REM Mixed mode so you can see
    what you're typing
300 END
400 &"CHART",TMD :REM Display Text page if there's
    an error
```

Now run the program. You can see that you are plotting data in line 200 in units that you have defined, the independent variable X varies from -3pi to +3pi, and the dependent variable Y ranges from +1 to -1. Once the SCL factor was set, AMPERCHART handles the rest. OK, now for some changes. Instead of a point plot, let's try a line plot. Make the following additions and changes.

```
170 &"CHART",MOV (-3*PI,SIN(-3*PI))
200 &"CHART",DRW (X,SIN(X))
```



Above is a picture of what you should now be seeing on your screen. If your screen appears to be different than the picture, check your program carefully. Pictures of the screen are interspersed into the text of this tutorial to provide a check for the reader.

Line 200 will now draw a line between the sequentially calculated points. The additional statement, line 170, establishes the starting point for the first DRW. It's not absolutely necessary, but without it, it's hard to say where the line will be drawn from when the first DRW command is executed in the FOR/NEXT loop. Try running the modified program first with statement 170, then remove it. See!

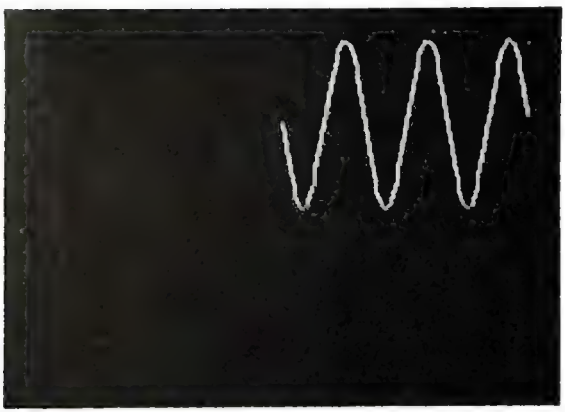
3.7 Clipping Windows

The SCL'ing function we've just covered obviously makes the plotting commands, PLT, DRW, and MOV, very useful. Now to sweeten the pie a little bit more, AMPERCHART goes one step beyond and amplifies the power of the SCL command with the ability to restrict your plotting to a user-defined window within the Hi-Res screen. This user-defined window could be equal to the Hi-Res screen. It could be 1/2, 1/4, 1/10, or whatever of the Hi-Res screen. The window can be square or rectangular; you determine it. This feature allows you to define and plot to as many non-interacting windows as you desire, within reason. Of course each window can have the same or different SCL'ing factor, it's all up to you, whatever you desire. All this flexibility is accomplished with the CLP (CLIP) command.

In order to define a working window with the CLP command you simply must specify the X coordinate of the left and right side of your window, and the Y coordinate of the bottom and top edges. That's it. Now for an example. We'll use the program we just typed in and modify it by defining a working clipping window in the upper right quadrant of the Apple screen. The X coordinates of the left and right sides of this quadrant are 140 and 279 respectively, the Y coordinates of the bottom and top are 96 and 191. Your window is thus invoked in the program by adding the following line:

```
120 &"CHART",CLP(140,279,96,191)
```

Now run the program. Instead of plotting to the entire screen, you should have 3 full-cycles of a sine-wave in a quadrant in the upper-right corner.



OK, so we've drawn a sine wave across a section of the screen...big deal, you could have done that with standard Applesoft. Right?...bite your tongue. For something a bit more difficult, we'll change to a new clipping window which is a small square somewhere near the center of the screen, and we'll display only the lower half of the sine wave.

The SIN(X) function varies from -1 to +1. If we set the SCL command so that the Y-axis goes from -1 to 0, only half the function will be within the clipping window and therefore only half of the function will be displayed. Although you won't see the upper half, AMPERCHART actually keeps track of the points that are outside of the window.

Here are the changes to be made:

```
120 &"CHART",CLP(90,190,50,150): REM A window in
    the center of the screen
160 &"CHART",SCL(-3*PI,3*PI,-1,0) :REM Just the
    lower half
```

Note that CLP is specified in screen coordinates, whereas SCL is specified in your user coordinates. The order of execution for CLP and SCL is not important; SCL always applies to the current clipping window.

3.8 Clears and Frame

After you run this simple program, and thought about what's been done and the speed at which it has been done, you'll have to agree that without AMPERCHART, this would be somewhat tedious and considerably slower with the standard Applesoft. On your own, experiment with different window sizes, ie. squares, rectangles, overlapping windows, etc, along with the RVS command.

You should also work with three other commands, CLR (CLeaR), FCL (Full CLear), and FRM (FRaMe). CLR will clear your currently defined window. It looks like it does the same thing as RVS, but it actually clears the window to the color black. RVS just compliments the color. FCL clears the entire Hi-Res screen, independently of what your current clipping window has been set to, and FRM draws a border around your current clipping window in the current HCOLOR. This is useful for highlighting your work area.

Before we go on, it should be mentioned, that the GGO command initializes HCOLOR=3, if you change HCOLOR, some of your PLT'ed points might not show up, and some vertically DRW'n lines might not appear. For now stick with HCOLOR=3.

There's one other point that should be mentioned regarding the variables initialized by GGO. Near the beginning of this tutorial, we did some simple plotting without having to define a CLP'ing window or a SCL'ing factor. The reason these didn't have to be specified is because the GGO command defaults the window and scale factor to mimic the standard Applesoft format. If you ever needed to get back to the standard Applesoft format (heaven forbid), you could do so with the following:

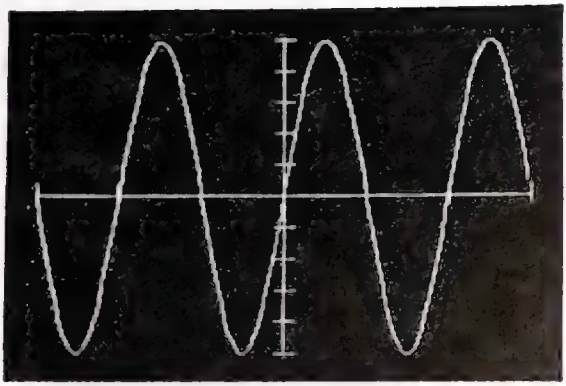
```
&"CHART",CLP (0,279,0,191)
&"CHART",SCL (0,279,191,0)
```

If you can't think of a little exercise right now that uses these commands, RUN TUTORIAL that's been included on the AMPERCHART Disk. This is a very simple program that will demonstrate most of what has been covered. Also take a look at MIND SCRAMBLER. This is an interesting 'nonsense' program that uses variously defined clipping windows to generate some unusual and captivating patterns. Experiment.

3.9 Axes and Grid Generation

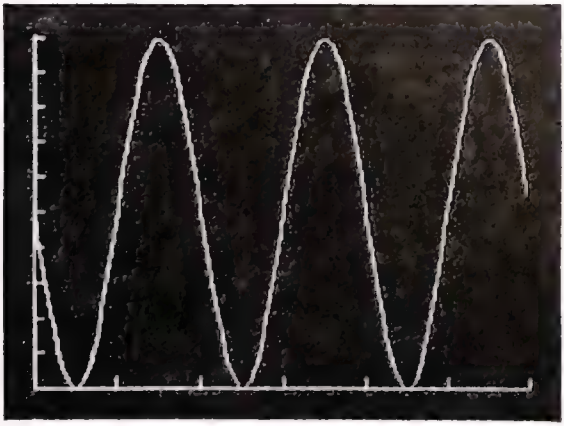
Now it's time for some 'tinsel and flash' in the data presentation. What's a plot without some type of axes?! AMPERCHART provides you with the ability to generate the usual X-Y axes within the clipping window, and a variety of labeling options. The command that handles the drawing of axes is AXS (AXeS). To use AXS, you must specify four variables. The first two are the separation, in user coordinates, between the tic marks on the X and Y axes. The second two variables are the X,Y coordinate where the two axes are to cross. Using the ORIGINAL program from pages 19-20 only, add the following statement:

```
165 &"CHART",AXS (PI, 0.2, 0, 0)
```



Run the program. The first variable, PI, specifies a tic mark every PI unit along the X axis: there should 6 tics altogether along the X-axis (remember X goes from -3pi to +3pi). Similarly the Y-axis goes from -1 to +1, and the second variable in the AXS command, 0.2, designates a tic mark every 0.2 units along the Y-axis, therefore there should be 10 tic marks. The last two variables were set to 0,0. This designates the origin (the crossing point of the axes) at X=0 and Y=0, which in this particular example is in the center of the clipping window. If you'd like your axes to start at the lower-left, try this:

```
165 &"CHART",AXS (PI, 0.2, -3*PI, -1)
```



There are some applications where there's a need to control the size of the tic marks, all the way from nothing to very large tic marks that simulate graph paper. Well AMPERCHART also allows you to specify the size of the tic marks for each axis, independently.

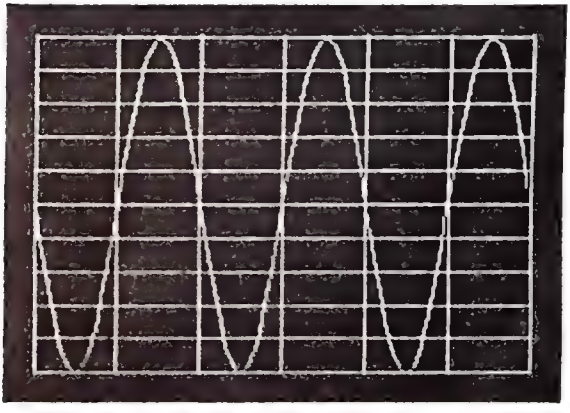
With the TSZ (Tic SiZe) command you specify the length of the X and Y tics in terms of the number of 'dots' each mark is to be. Try the following statement:

```
163 &"CHART",TSZ (0,0)
```

You should see your axes drawn without any tic marks at all. To go to the other extreme set your TSZ to anything greater than 279 (remember that values greater than the clipping window are clipped).

```
163 &"CHART",TSZ (300,300)
```

Now you've got graphpaper!

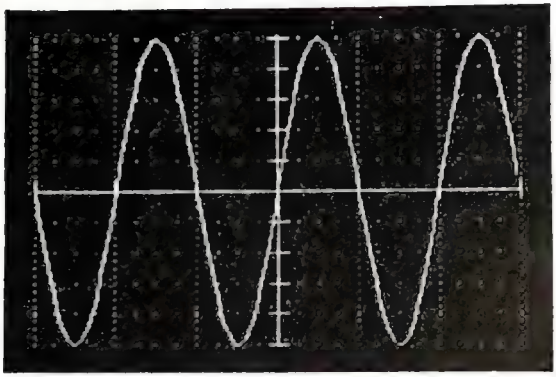


AMPERCHART also offers another command that is similar to AXS, except that dots are plotted at the intersection of the imaginary grid lines instead of tic marks being drawn along the axes. The variables specified by this other command, GID (GrID), are essentially the same as AXS. The separation between adjacent grid lines is determined the same way as X-tic and Y-tic were in the AXS command, as is the intersection of the X and Y axes. To demonstrate the GID command, make this change:

```
165 &"CHART",GID (PI, 0.2, 0, 0)
```

If you'd like to try for an interesting effect you can get with GID, try the following, and think about what we've done.

```
163 &"CHART",AXS (PI, 0.2, 0, 0)
165 &"CHART",GID (PI, 0.05, 0, 0)
166 &"CHART",GID (PI/4, 0.2, 0, 0)
```

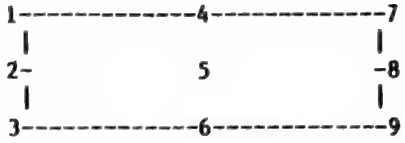


You can see that AMPERCHART offers you a very flexible formatting style for your data presentation.

3.10 Labeling

A graphics presentation is not complete without proper labeling of the axes, legends, etc. AMPERCHART provides you with a very flexible label generation and positioning capability. There are two commands, HLB (Horizontal LaBel) and VLB (Vertical LaBel), which enable you to place alphanumeric labels on your screen. Each command requires that you specify the alphanumeric string, the X,Y user coordinates at which you desire the label to be placed, and a label position code (more on this later).

The alphanumeric string can either be a literal string, i.e. "WEIGHT PER UNIT VOLUME", or a string variable or expression, A$ or STR$(X) for example. The position code can be a little difficult to comprehend the first time it's introduced, so don't hesitate to go through this section slowly. Try to visualize your label as a simple rectangular object with four sides, four corners, and a center. Each of the sides, corners, and center is assigned a number, as seen here:



These position codes are used to determine how your label is placed relative to the specified X,Y coordinates. For example the position code '7' refers to the top-right corner of your label. When you use the position code '7' with your label, AMPERCHART will position your label such that the top-right corner is positioned at the X,Y coordinate you specified with the VLB or HLB command. Similarly, the position code '5' would place your label dead-center on the specified X,Y coordinate.

Now for some real examples. First we'll reinitialize AMPERCHART, position the cursor at the bottom of the Text screen, set Mixed-mode Graphics, and set the SCL factor so that the screen goes from 0-100 along both axes.

From the immediate mode (i.e. not as part of your program), type in:

```
VTAB 23
&"CHART",GGO(1)
&"CHART",MIX
&"CHART",SCL(0,100,0,100)
```

The center of the screen is at 50,50. Put a dot there, so we'll have a visual marker.

```
&"CHART",PLT(50,50)
```

Now that you can see coordinate (50,50), we'll put a label there, first with a position code '7' and then position code '5'. For the sake of this demonstration, we'll be real original and use the string "LABEL" as our label.

```
&"CHART",HLB("LABEL",50,50,7)
```

The string "LABEL" should now be positioned such that the plotted point at 50,50 is in the top-right corner of the imaginary rectangle encompassing the letters in "LABEL". OK, now we'll try position code '5'. This time we'll use a string variable instead of a string literal.

```
&"CHART",CLR          :REM Clear the screen
&"CHART",PLT(50,50) :REM Replace the dot
A$="LABEL"            :REM Define the string variable
&"CHART",HLB(A$,50,50,5)
```

The string should now be positioned such that the middle of the letter "B" is on top of the dot you plotted. The VLB command performs identically, except the string is printed from the bottom to the top. And for the purposes of the position code, you should visualize the label as a rectangle taller than it is wide.

As with all the other commands that plot or draw on the screen, your label will be clipped if drawn outside the window. It's been found though that when axes are drawn, they are often times drawn on the edge of the clipping window, so axes labeling, with clipping in effect, is sometimes awkward. To eliminate this possible problem, AMPERCHART provides the user with the ability to place labels outside the window if desired. The clipping function can be bypassed for the VLB and HLB commands simply by putting a minus sign in front of the position code.

If you want to erase a label that you've drawn on the screen (for example, to print a new prompt or other phrase), just set HCOLOR to black (or whatever your background color is), and then do a HLB or VLB again at the original position.

There's a program on the Chart 'n Graph Toolbox disk called HI-RES CHARACTER DEMO that displays the currently available character set. If you request a character outside the allowed range e.g. a control character, it will result in a default character (x) being printed. As an aside, you'll notice this program displays a flashing cursor on the Hi-Res screen. Look at how it's done.

3.11 Display and Working

Up to now, we have been working solely with Hi-Res Page 1, but there are two Hi-Res pages available on the Apple II. You could have specified Page 2 in the initialization procedure (&GGO(2)), but special precautions must be taken before using page 2. Specifically, this part of memory (page 2) must be protected by using RELOCATE II.TB instead of RELOCATE I.TB. With RELOCATE II.TB, both Hi-Res pages are available for your program. This does take an extra 8K of memory though, so if you don't really need page 2, stick to the usual approach.

If you are using both pages, it would be nice to be able to switch from one Hi-Res page to another. AMPERCHART provides the ability to do just this, with the DSP (DISPlay) command, with which you simply request which page you'd like to see.

AMPERCHART also provides you with the ability to display one Hi-Res page and yet work, i.e. PLT, DRW, AXS, HLB, etc. on the other page, if you want. You specify the 'working page' with the WRK (WoRK) command. There are a number of applications for these two commands, one of the most obvious is something like animation. This is a process where the user DSP's one page while WRK'ing on the other. When the plotting process is complete, the DSP page is switched and the previously DSP'ed page is then re-WRK'ed. There is an example of this type of page switching program on the AMPERCHART Disk, under the name LITTLE ANIMATION. Run it, and then examine the listing to see what was done.

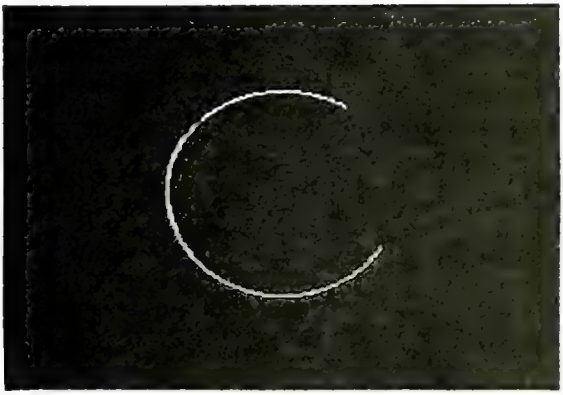
3.12 Arc Draw

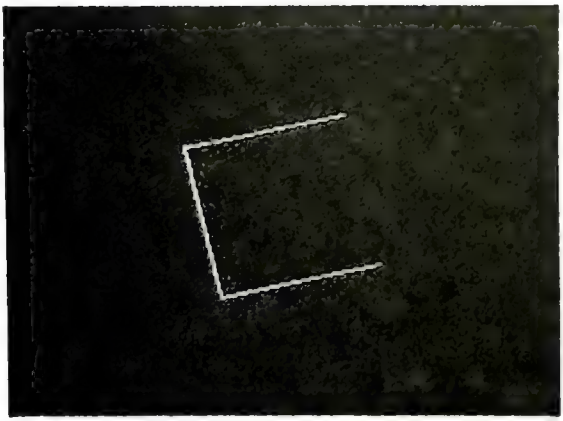
So far we have concentrated on x-y plotting. Some information is better illustrated using a pie chart, so we've included a function which will make pie charts a cinch, the ARC command. Enter and then RUN the following program:

```
  1  CALL PEEK(175) + 256 * PEEK(176) - 46
  2  &"RELOCATE"
 10  &"CHART",GGO(1) : REM INITIALIZE GRAPHICS
 20  &"CHART",MOV(140,96) : REM MOVE TO CENTER OF
     THE SCREEN
 30  &"CHART",ARC(60,2,360,270) : REM DRAW PART OF A
     CIRCLE
100  GET A$ : &"CHART",TMD
```

If all worked OK you should see 3/4 of a circle on the screen. The variables in the ARC command are in order: the radius of the figure to be drawn, the starting angle in radians (´0´ radians is at the top of the screen), the number of sides in the total figure, and the number of sides to be drawn. The ARC command in fact draws polygons, but on the Apple´s screen a many sided polygon appears as a circle. To demonstrate this try changing line 30 to:

```
30 &"CHART",ARC(60,2,4,3)
```





ARC draws in a clock-wise fashion, unless the radius variable is set to a negative value. You can also get spirographic patterns by setting the number of sides to non-integer amounts.

Getting back to the pie chart, change line 30 to:

```
30 &"CHART",ARC(-60,2,360,270)
```

Upon running this you'll find that suddenly the end points of the arc are connected to the center. This is an option of the command that is selected by making the radius negative. There are other options for this command that are explained in the Command Reference chapter.

One other note. If you prefer your circles defined in degrees, rather than radians, just multiply your 'degree' value by 0.01745 right in the arc statement:

```
30 &"CHART",ARC(60,DEG*.01745,4,3)
```

3.13 Area Filling

To finish up the pie chart add the next two lines:

```
40 &"CHART",ARC(-60,2,-360,90)
50 &"CHART",FIL(140,80)
```

Now the program will draw the last piece of the pie (notice that it gets drawn backwards; that is caused by the negative number of sides).

The command in line 50 is a new command: FILL. It simply tries to fill in the area surrounding the location that the user selects.

If you'd like to experiment with more complete fill algorithms or would like to read the screen for some other purpose, there is a command called BIT. It returns the color of any selected screen coordinate. It's probably not something you'll use everyday, but it can be handy especially for writing a custom screen dump routine.

3.14 Logarithmic Mode

This command (LMD) is principally for scientists or engineers (although we don't mind if anyone else wants to make use of it too!). The Log MoDe command allows you to force the scaling and axes generation to be logarithmic in either or both coordinates. The log mode defaults to linear-linear plotting, but when set to lin-log or log-log modes, it affects most of the other commands in the package. See the command reference section, the LMD Tutorial on the Chart 'n Graph Toolbox disk, and the demos for more information.

3.15 Screen-to-User Coordinate Conversion

The function of this command (STU for "Screen To User") is to convert screen coordinates (Apple's coordinates 0-279 and 0-191) to user coordinates. If you're scratching your head on this one, that's OK. One of the primary purposes of AMPERCHART is to allow the user to think about the screen in terms of his/her coordinate system without worrying about the Apple's 0-279 and 0-191 system, a job it does very nicely. But, sometimes its necessary to provide a link between the user coordinate system and the Apple's modified coordinate system (remember Y=0 is on the bottom), and the STU provides this link.

As an example, suppose you are drawing a number of plots on the screen and you want to place a descriptive legend somewhere near the upper-right of the screen. In order to determine the coordinates of the area where you want to place your legend, it is probably easier to think about the position in terms of the Apple's coordinate system. With this example, the center of the legend is in the region of X=250 and Y=175 (screen coordinates). You would then pass the screen coordinates to the STU command, which then returns with the equivalent user coordinates. The returned user coordinates could then be used in HLB or VLB command to put your legend on the screen.

Take a look at LABEL TEST. You'll see that the STU command was used at the beginning of the program to help position the alphanumeric prompt on the screen.

3.16 Location Command

The final command, LOCation, returns the current location of the AMPERCHART program in memory. When AMPERCHART is used in conjunction with the Workbench, it tends to move around in the machine (if you listen carefully you may hear it sloshing around) and the current location can be handy for fiddling with AMPERCHART innards. A table of interesting locations is given in Chapter 11.

DEMONSTRATION PROGRAMS

There are a number of demonstration programs on the Chart 'n Graph Toolbox disk that do a variety of interesting things. They are there for you to run, examine, ponder, modify, and learn. You should have already looked at MIND SCRAMBLER, LABEL TEST, and CHARACTERS. If you haven't, take a look at them now. They are relatively short, but illustrate the variety of techniques which can be performed using AMPERCHART.

There are two programs that make extensive use of bar-charts, BAR CHART and FINANCIAL. Look at the two different techniques used to generate the barcharts. Finally in SAMPLE FLOW CHART, you'll find a somewhat unusual application of AMPERCHART that goes beyond the usual data presentation format of X-Y plots. Though once you think about it, AMPERCHART is an ideal tool for this type of application.

CONCLUSION

That concludes the formal presentation of this tutorial. It is hoped that you now have an appreciation and understanding of the utility and power of the AMPERCHART command set. For additional practice try running the program "TUTORIAL" on the Chart 'n Graph Toolbox diskette. You should be able to incorporate AMPERCHART, with very little effort or difficulty, into almost any graphics-intensive program, making your job many times easier.

4. AMPERCHART COMMAND REFERENCE

AMPERCHART is a library of 'EXTRA' graphics related Applesoft commands. They can be used on any 48k Apple II with ROM Applesoft or the Language System, and one disk drive (DOS 3.3).

Each of the 27 commands performs a single function which give the programmer the building blocks needed to design his own plotting package, as simple or complex as is needed.

The names of the new commands are:

GGO	TMD	GMD	MIX	NMX	DSP	WRK
CLP	SCL	LMD	CLR	FCL	RVS	MOV
PLT	DRW	BIT	AXS	GID	HLB	VLB
FIL	ARC	TSZ	FRM	STU	LOC	

Each command will now be discussed in detail. Examples will follow the descriptions.

Command: GGO(page)

Meaning: Graphics GO

Purpose: Initialize graphics
(similar to HGR & HGR2)

Parameters: page - The number (1 or 2) of the Hi-Res graphics page to be used.

Results: The selected (primary or secondary) Hi-Res screen is displayed.
Full screen graphics is set (no text at bottom).
The screen is cleared to Black1.
HCOLOR is set to 3.
The coordinate (0,0) is redefined to be in the lower left hand corner of the screen.
The clipping window is set for full screen use (0-279,0-191).
The screen is scaled in X from 0-279 and in Y from 0-191.
Both axes are set to linear mode.
The X-axis tic size is set to 2 dots and the Y-axis tic size is set to 3 dots.

Remarks: This command is normally used to set and initialize the graphics mode.

Example: 10 &GGO(1)

Command: TMD

Meaning: Text MoDe

Purpose: Display text screen (similar to TEXT)

Parameters: none

Results: The primary text page is displayed.
The scrolling variables of the text window are not reset (unlike TEXT).

Example: 20 &TMD

Command: GMD

Meaning: Graphics MoDe

Purpose: Display last used Hi-Res graphics screen

Parameters: none

Results: The last used (either primary or secondary) Hi-Res graphics page is displayed.

Example: 30 &GMD

Command: MIX

Meaning: MIX

Purpose: Display graphics and bottom four lines of text

Parameters: none

Results: If in graphics mode, bottom four lines of text page (either primary or secondary) are displayed.
If in text mode, switch is set so that the bottom four lines of text will be displayed when ´&GMD´ is executed.

Remarks: The Y range visible in this mode is 32-191 in screen coordinates.

Example: 40 &MIX

Command: NMX

Meaning: No MiX

Purpose: Display full screen as graphics (opposite of MIX)

Parameters: none

Results: If in graphics mode, full screen will be displayed as graphics.
If in text mode, full screen will be displayed as graphics when ´&GMD´ is executed.

Example: 50 &NMX

Command: DSP(page)

Meaning: DiSPlay

Purpose: Display either primary or secondary page independently of which one is being used

Parameters: page - The number (1 or 2) of the page to be displayed.

Results: If in graphics mode, the selected (either primary or secondary) Hi-Res graphics page is displayed.
If in text mode, the selected (either primary or secondary) text page is displayed.

Remarks: In conjunction with the WRK command (below), allows smooth animation by displaying one page while clearing and drawing on the other.

Example: 60 &DSP(2)

Command: WRK(page)

Meaning: WoRK

Purpose: Define which Hi-Res screen is to be worked on independently of which screen is displayed

Parameters: page - The number (1 or 2) of the Hi-Res graphics page to be worked on.

Results: All clearing, drawing, and labeling is defined to take place on the specified screen, (primary or secondary) whether it is displayed or not.

Example: 70 &WRK(1)

Command: CLP(sxmin,sxmax,symin,symax)

Meaning: CLiP

Purpose: Define a clipping window, as a subset of the Hi-Res screen, in which all plotting commands will be confined.

Parameters:
sxmin - left side of the clipping window in screen coordinates (0-278).
sxmax - right side of the clipping window in screen coordinates (1-279).
symin - bottom of the clipping window in screen coordinates (0-190).
symax - top of the clipping window in screen coordinates (1-191).
NOTE: sxmin<sxmax & symin<symax

Results: Only points, lines and (if desired) labels that fall within this window will be displayed.

Remarks:
No errors result from trying to work outside this window.
The previous user scaling specified (or the default) now is applied to this window.
When the clipping window is modified, the very next point or line drawn may not be clipped. This was a design choice made to improve average plotting speed. To avoid this problem perform a move before plotting or drawing after a clipping change.

Example: 80 &CLP(50,229,32,191)

Command: SCL(uxmin,uxmax,uymin,uymax)

Meaning: SCaLe

Purpose: Define the current clipping window's range in user coordinates

Parameters:
uxmin - user's defined value for the left side of the current clipping window.
uxmax - user's defined value for the right side of the current clipping window.
uymin - user's defined value for the bottom of the current clipping window.
uymax - user's defined value for the top of the current clipping window.
NOTES: Uxmin must not equal uxmax
Uymin must not equal uymax
If log mode is set, those variables along the log axis or axes must be greater than 0.

Results: All move, plot, draw and label commands are made in these new user defined coordinates.
This scaling remains in effect until another SCL is issued or graphics is re-initialized with a GGO.

Remarks: Reverse scaling can be defined by setting uxmin less than uxmax and/or uymin less less than uymax.
Scaling defaults to linear, but can be made logarithmic by the LMD command.

Example: 90 &SCL(-100,100,-50,150)

Command: LMD(mode)

Meaning: Log MoDe

Purpose: Define X and Y axes to use either linear or log scaling

Parameters: mode - log/linear mode (0-3)
default 0

Results: LMD(0) sets both X and Y scaling as linear
LMD(1) sets X log and Y linear scaling
LMD(2) sets X linear and Y log scaling
LMD(3) sets both X and Y scaling as log

Remarks: For log scaling, both the minimum scaling and the maximum scaling value must be greater than 0, therefore the default scaling of (0,279,0,191) would cause an error if LMD was set to 1,2 or 3.
Many other commands including MOV, PLT, DRW, AXS, and GID are affected by this setting.
Certain combinations may produce strange results, e.g. circles drawn in log mode are no longer circles.

Example: 100 &LMD(3)

Command: CLR

Meaning: CLeaR

Purpose: Clears current clipping window

Parameters: none

Results: Only current clipping window is cleared to Black1 (bit 7 is cleared).

Example: 110 &CLR

Command: FCL

Meaning: Full CLear

Purpose: Clears full screen

Parameters: none

Results: Clears whole screen to Black1 (bit 7 is cleared).
All clipping and scaling variables remain unchanged.

Remarks: This is a fast clear - faster than the one given for Applesoft making it useful for animation.

Example: 120 &FCL

Command: RVS

Meaning: ReVerSe

Purpose: Reverse everything in the current clipping window

Parameters: none

Results: All colors are reversed in the current clipping window (bit 7 is unchanged).
Everything else on the screen is unchanged.

Remarks: Can be used to simulate a blinking cursor by alternatively reversing a given area.
Also useful for highlighting titles.

Example: 130 &RVS

Command: MOV(ux,uy)

Meaning: MOVe

Purpose: Move to a user specified point without line generation

Parameters: ux - user X coordinate
uy - user Y coordinate

Results: Establishes a starting point for the next line to be drawn.
Display remains unchanged.

Example: 140 &MOV(-50,-35.7)

Command: PLT(ux,uy)

Meaning: PLoT

Purpose: Plot a point at the coordinate specified by the user

Parameters: ux - user X coordinate
uy - user Y coordinate

Results: Leaves a point, in the current HCOLOR, at the user scaled coordinate.
Will not be displayed if outside clipping window.
Establishes starting point for line generation.

Example: 150 &PLT(79,140)

Command: DRW(ux,uy)

Meaning: DRaW

Purpose: Draws a line to user specified coordinate

Parameters: ux - user X coordinate
uy - user Y coordinate

Results: A line is drawn from the last point moved to, drawn to, or plotted at, to the user scaled coordinate.
Line is drawn in current HCOLOR.
Any parts of the line falling outside the clipping window are not displayed.

Example: 160 &DRW(200,300)

Command: BIT(sx,sy,color)

Meaning: BIT color

Purpose: Determine color of bit at screen location sx,sy

Parameters:
sx - screen X coordinate of bit (0-279)
sy - screen Y coordinate of bit (0-191)
color - color of bit (must be a variable)

Results: Color of bit at sx,sy is returned. Value will be one of (0,1,2,4,5,6).
Note: white1 and white2 cannot be determined.

Remarks: To convert user to screen coordinates see "LOCATIONS OF INTEREST" below.
Can be used to write custom screen dump routine.

Example: 170 &BIT(12,118,C)

Command: AXS(xtic,ytic,xorigin,yorigin)

Meaning: AXeS

Purpose: Draw an X-Y axes in clipping window

Parameters:
xtic - distance between X-axis tic marks in user coordinates.
ytic - distance between Y-axis tic marks in user coordinates.
xorigin - X coordinate of Y-axis.
yorigin - Y coordinate of X-axis.

Results:
An X-Y axes is drawn in the current clipping window in the current HCOLOR.
When in log mode, the xtic and/or ytic variable(s) specify that the tic marks are to be placed at multiples of 10 divided by the tic mark variable. Thus, if LMD(1) is in effect and xtic=10 is specified, there will be nine tic marks in each decade located at multiples of (1,2,3,4,5,6,7,8,9). If xtic=5 is specified there will be five tic marks per decade located at multiples of (1,2,4,6,8). xtic=2 will specify tic marks at multiples of (1,5). The four log tic mark variable values (1,2,5,10) are generally the most useful since they are the only values of the variable that specify integer tic mark locations.
When in log mode, the xorigin and yorigin variables are the same as previously defined.
Resets starting point for DRW.

Remarks:
The tic mark lengths can be changed by the TSZ command (default:2 for X & 3 for Y).
If HCOLOR=1,2,4, or 5, some lines may not be displayed (adjust clipping window or use White1 or White2).
Close tic marks may not appear evenly spaced due to the limited screen resolution.

Example:
```
180 &AXS(10,10,0,0)
```

Command: GID(xtic,ytic,xorigin,yorigin)

Meaning: GrID

Purpose: Draw an axes with grid lines in clipping window

Parameters:
xtic - distance between X-axis grid lines
ytic - distance between Y-axis grid lines
xorigin & yorigin - same as AXS

Results:
An X-Y axes is drawn in the current clipping window in the current HCOLOR.
The dots in the grid lines intersect at the xtic & ytic intersection points.
The effect of log mode is the same as for AXS.
Resets starting point for a DRAW.

Remarks:
As with the AXS command, axes and dot visibility depends on HCOLOR.
Can be used twice to get widely spaced dotted lines along both axes.

Example: 190 &GID(10,10,0,0)

Command: HLB(string$,ux,uy,position)

Meaning: Horizontal LaBel

Purpose: Place a horizontal label at the requested user coordinate

Parameters:
string$ - any string
ux - user X coordinate
uy - user Y coordinate
position - positioning parameter

Results:
Horizontal label is placed at the specified coordinate in current HCOLOR.
For a positive positioning parameter, only those characters completely falling inside the current clipping window are displayed.
A negative positioning parameter allows the label to be positioned outside the window but will be clipped to screen boundaries.
Legal characters are ' '...'_'
(ASCII 32-95).
The positioning parameter is explained below.

Example:
```
200 &HLB("ANY STRING YOU WANT.",5,-7,5)
```

Command: VLB(string$,ux,uy,position)

Meaning: Vertical LaBel

Purpose: Place a vertical label at the requested user coordinate

Parameters: (same as HLB)

Results: Results from HLB apply except the label is written vertically.
The label is written from the bottom, up (top of the characters facing left).

Example: 210 &VLB("LABEL YOUR Y-AXIS.",-30,0,2)

The positioning parameter allows accurate placement of the label relative to the chosen X-Y coordinate. The values are:

```
1  4  7
2  5  8
3  6  9
```

Think of these as positions about the label. A '1' means place the X-Y coordinate in the upper left hand corner of the label, a '5' places the X-Y coordinate at the center of the label, etc. See the examples and experiment!

Command: FIL(ux,uy)

Meaning: Fill a black convex area with the current HCOLOR

Parameters: ux - user X coordinate
uy - user Y coordinate

Results: Area surrounding ux,uy will be filled with current HCOLOR.
ux,uy must be either black1 or black2 and on the screen (in the current clipping window) to be filled.

Remarks: Only figures containing no angles greater than 180 degrees can be correctly filled in general; occasionally, edges of objects will not be filled - this is due to the stair stepping of the raster display creating non-convex edges.
For best results draw outline of area to be filled in white.

Example: 220 &FIL(PI/2,.5*SIN(PI/2))

Command: ARC(radius,angle,sides,number)

Meaning: ARC

Purpose: To generate arbitrary sections of regular polygons, including circles (use a large number of sides).

Parameters:
radius - radius of polygon in user coordinates
angle - starting angle of arc in radians (where angle=0 is straight up)
sides - total number of sides in the polygon
number - of the total sides, the number to be drawn

Results: A subset of a regular polygon is drawn in the current user coordinates, centered at the last point plotted or moved to.
A negative radius causes the beginning point on the arc and the ending point on the arc, to be connected to the center. This is useful for pie charts. A negative value for sides causes the arc to be drawn in a counter-clockwise direction. A negative value for number causes only the verticies of the polygon to be plotted. This is useful for building polar grids.

Remarks: The utility of this command stems from its many modes.
Noninteger values for sides can give strange but useful results (stars,etc.).
The starting angle can be in degrees if that value is multiplied by 0.01745 as part of an expression as in:

ARC(10,DEG*0.01745,360,180) or:

ARC(10,45*0.01745,360,180)

Note that normal Applesoft's SIN,etc. functions assume a counter-clockwise direction, with an angle of 0 radians being the far right edge of the screen.

Example: 230 &ARC(90,PI/2,2.5,5)

Command: TSZ(xsize,ysize)

Meaning: Tic SiZe

Purpose: Adjust the axes' tic mark sizes

Parameters: xsize - size of X-axis tic marks in dots.
ysize - size of Y-axis tic marks in dots.
(range of -32767 to +32767)

Results: Lengths of tic marks are set for next AXS command.
Default xsize=2, ysize=3.

Remarks: Use tic mark sizes greater than 279 for 'graph paper' effect.

Example: 240 &TSZ(5,5)

Command: FRM

Meaning: FRaMe

Purpose: Outline current clipping window

Parameters: none

Results: The current clipping window is outlined in the current HCOLOR.

Remarks: Depending on HCOLOR, some vertical lines may not be visible (adjust window).
Frame is just inside window.

Example: 250 &FRM

Command: STU(sx,sy,uxv,uyv)

Meaning: Screen To User

Purpose: Convert screen coordinates to those of the user

Parameters: sx - screen X coordinate (-32767 to 32767)
sy - screen Y coordinate (-32767 to 32767)
uxv - user X coordinate
(must be a variable).
uyv - user Y coordinate
(must be a variable).

Results: Procedure converts screen coordinates into user scaled coordinates and returns the values in uxv & uyv.

Remarks: Use to position labels in screen units.
Find maximum and minimum user values to avoid 'ILLEGAL QUANTITY ERROR'.
Many other uses.
For inverse function see "LOCATIONS OF INTEREST" below.

Example: 260 &STU(140,96,X,Y)

Command: LOC(address)

Meaning: LOCation

Purpose: Return current location of AMPERCHART

Parameters: address - starting address of AMPERCHART (must be a variable)

Results: Address returned is used with offsets to make run-time modifications to AMPERCHART.

Example: 270 &LOC(X)

5. OTHER LIBRARY FILES

5.1 Use With and Without the Workbench

This Chapter describes other useful commands which have been included on the AMPERCHART Diskette. These commands are completely Workbench compatible (see Chapter 2 for a description of the Workbench), and this is the (highly) recommended way of using them, but they can also be used as stand alone commands.

The use of any combination of these commands in a stand alone configuration involves a three-step process:

1. The commands must be loaded into memory in some location that will not interfere with other programs,

2. The commands must be protected from being destroyed by standard Applesoft functions, and

3. The commands can then be called from within a program by using a special syntax.

In general, it is best to place these commands just below the top of available memory. If AMPERCHART is not being used this corresponds to $9600. If a version of AMPERCHART is installed at the upper end of memory, you are using a MAXFILES value other than '3', or you have a utility such as G.P.L.E. or Apple's RENUMBER utility installed, then the highest available location can be found (in decimal) with the following statement:

```
10 HI = PEEK(115)+256*PEEK(116)
```

Once the highest available location has been determined, simply load in the command (or commands) just below this location, e.g.

```
15 HI = HI - 281
20 PRINT CHR$(4);"BLOAD HI/LO ZOOM.TB,A";HI
```

where HI is the address obtained by subtracting the length of the command you are loading (in this case 281 bytes) from the address of the highest available location in memory.

Next we must protect the command(s) by moving the HIMEM pointer down below the address of the beginning of the command(s). This can be done by executing a HIMEM:HI instruction, where HI is the address of the beginning of the commands (the same as the address that they were loaded at above.) This is exactly the technique that is used in Section 4.3 to illustrate the use of AMPERCHART in a stand alone configuration.

For example, the HI/LO ZOOM.TB command could be installed within an Applesoft program by the following instructions:

```
10 HI=PEEK(115)+256*PEEK(116) : REM Read HIMEM
20 HI=HI-281 : REM Adjust for length of command
30 PRINT CHR$(4);"BLOAD HI/LO ZOOM.TB,A";HI
40 HIMEM : HI
```

Finally, the calling syntax must be modified from that shown in the following pages. The syntax which is shown is for use with the Workbench only. When used in a stand alone mode the calling syntax becomes:

```
CALL (HI) variable list
```

where HI is the beginning address of the command and variable list is the set of variables specified in the manual. Note that when Workbench is not being used to add the commands, the CALL address is in parentheses and no comma precedes the variable list.

For the example above, a typical call from within the program would be:

```
200 CALL (HI) X,Y,0
```

This technique is further illustrated in the demo program HIRES ZOOM/MEDIAN FILTER DEMO.

>> RELOCATE I & II.TB <<

AND

>> AMPERCHART JUMP FILE.TB <<

by Rick Chapman

FUNCTION: RELOCATE I.TB and RELOCATE II.TB automatically load AMPERCHART.TB and optimize memory usage on the Apple. The module AMPERCHART JUMP FILE.TB is then used as the link to the loaded AMPERCHART.TB module.

LENGTH:

RELOCATE I.TB: 303 bytes ($12F)
RELOCATE II.TB: 303 bytes ($12F)
AMPERCHART JUMP FILE.TB: 3 bytes ($3)

SYNTAX:

For RELOCATE: &"RELOCATE"
For JUMP FILE: &"CHART",Command [,Parameter list]

HOW TO USE IT: RELOCATE I.TB is used as one of the very first statements in a program that uses AMPERCHART.TB and Hi-Res page 1. If use of page 2 is desired, RELOCATE II.TB should be used. AMPERCHART.TB must be on the diskette in the active drive at the time the program is run so that RELOCATE can properly load it.

The AMPERCHART JUMP FILE.TB module is appended to the user program and given the invocation name desired for the AMPERCHART command package.

See Chapter 7 for more information on alternative setups. The jump file itself is nothing more than a JMP $800 instruction. For a description of the uses of jump files, see Appendix B.

Here's a illustration of how memory is set up and he JUMP FILE operates:

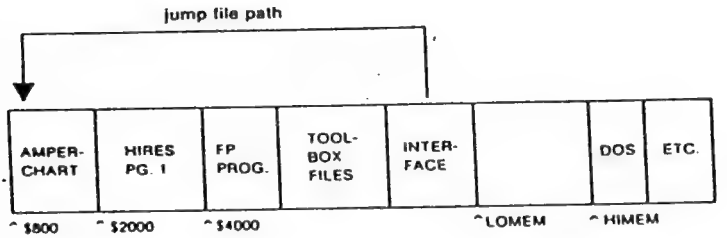


Diagram of AmperChart Jump File Operation
(AMPERCHART.TB loaded by RELOCATE I.TB)

LIMITATIONS: AMPERCHART.TB must be available for the RELOCATE routines to load from disk. RELOCATE I.TB and RELOCATE II.TB both use memory locations $2A0 through $3CF (inclusive).

>> SPLITTERTB

by Rick Chapman

FUNCTION: Splits any Applesoft program, with any machine language routines that may be attached to it, around the high-resolution graphics pages. This allows long programs to make use of Hi-Res graphics without overwriting themselves. An unsplit function is also provided to aid in the development of programs that use the splitter.

LENGTH: 822 bytes ($336)

SYNTAX: &"NAME",splitflag
&"NAME",aexpr

SAMPLE: &"SPLIT",0

=> Unsplits program. If the program was already unsplit, it has no effect.

&"SPLIT",1

=> Splits program around Hi-Res page 1. If the program was already split, or resides above Hi-Res page 1, an error occurs.

&"SPLIT",2

=> Same as above, but splits around Hi-Res page 2.

&"SPLIT",3

=> Same as above, but splits around both Hi-Res pages.

HOW TO USE IT: This routine is only needed if RELOCATE I.TB or RELOCATE II.TB is not being used. See the SPLITTER DEMO for a graphical illustration of how SPLITTER is used.

The high resolution graphics in the Apple][is bit mapped. This requires two adjacent areas of the Apple's memory to be set aside for the two Hi-Res screens or pages. Due to technical reasons, the areas chosen for the Hi-Res pages are in the middle of the available program memory.

Unless special procedures are followed, programs longer than 6K (the total of both the Applesoft program and any attached machine language code), will reside at least partly in one of these areas, making it impossible to use the Hi-Res graphics. Programs can be loaded into the memory above the Hi-Res pages, but this typically wastes the 6K of available memory below the Hi-Res pages.

This routine allows your programs to split themselves around either or both of the Hi-Res areas. The procedure to do this is rather involved, but the routine takes care of all of the details. Once split, most programs will run normally and can even be listed without problem. On the other hand, editing of the split program (inserting or deleting lines) will cause the apparent loss of the portion of the program above the gap. Never fear though, the unsplit function takes care of this problem so that you never have to worry about destroying your program.

The only restriction on running split programs is that DATA statements occurring after the split (which will occur far into the program) will be ignored by READ statements. The only fix is to place all DATA statements near the beginning of the program.

When split is called, there are several different types of results depending on the length and type of programs currently occupying memory.

If the program envelope, that is the Applesoft program and all of its attached Toolbox commands, is entirely below the selected Hi-Res area, then nothing is moved, but the internal beginning of variable and array pointers are set above the Hi-Res page. This prevents the program variables and arrays from entering into the Hi-Res area. It also means that you do not need to execute your own LOMEM: command to protect a Hi-Res page.

If the Applesoft program lies entirely below the selected Hi-Res area, but the attached machine language routines fall within the Hi-Res area, then only the machine language routines are moved. Again the variable and array pointers are updated, this time to the address end of the newly located machine language routines. In addition, the ampersand vector is updated to the new locations, so that calls to the machine language subroutines will still work.

If the Applesoft program lies partly within the Hi-Res area, then the Applesoft program is split and the end of the program along with any attached machine language routines is moved up beyond the Hi-Res area. The split program will still run and list just as it did before, but the internal structure of the program has changed resulting in certain unavoidable side-effects.

The principal side effect is that editing of the program will cause the second half of the program to apparently disappear, and in addition a mysterious new line will suddenly appear at the very end of the remaining program; a GOTO that references itself. This new instruction is the key to a sucessful split, but is normally hidden from view. When the program was edited, the links between the beginning and ending sections of the program are temporarily lost and this hidden line appears. A program in this form will not execute succesfully.

To recover from this apparent catastrophe, you simply have to reconnect the ampersand vector by executing only the first line of the program (just list the first line, pass over it with the cursor, and then hit return) and then type &"SPLIT",0. An alternative method of reconnecting the ampersand vector is available; insert the AMPERCHART diskette and type:

```
EXEC RECONNECT
```

This simple procedure will remove the new GOTO, unsplit the program, and correctly reset all of the pointers. Do not delete the GOTO or add any lines after it before you unsplit; the unsplitter portion of the code looks for that GOTO and if it isn't there the results are unpredictable. Otherwise, the unsplit will successfully and reliably repair the program to its original condition.

This effect also occurs if you try to save a split program to disk. Recovery after loading the program can be made in the same manner as for the edit case. Note that the split program will take up 21 to 41 extra sectors on the disk than the unsplit version.

It is recommended that an unsplit be performed before each split, to avoid errors and to insure the integrity of the program. For most applications, it is best to include the following three lines in each program in which the splitter will be used.

```
1 CALL PEEK(175)+256*PEEK(176)-46:REM AMPERSAND SET UP
2 &"SPLIT",0 : REM UNSPLIT
3 &"SPLIT",1 : REM SPLIT
```

When using the splitter in this manner it is important that the split occur within the program BEFORE any variables are used, as the splitter does not move or preserve variables.

All of the above examples assume that the splitter will be used in conjunction with the Chart 'n Graph Toolbox. The splitter can also be used by itself, or with other machine language routines without the aid of Routine Machine. The calling format is modified when operating stand alone to:

```
&0
&1
or
CALL (N)0
```

where the address N shown in the call is the location that the splitter was loaded into.

When operating the splitter stand alone, the ampersand vector is is not modified by the splitter, unless the ampersand vector pointed into the program envelope before the split. Thus, the splitter takes care of most contingencies arising from its use in various modes and memory locations.

The following error codes are displayed:

```
ILLEGAL QUANTITY   : splitflag<0 or >3
CAN'T CONTINUE     : trying to split a split program
UNDEFINED FUNCTION : split program begins above the gap
OUT OF MEMORY      : split requires too much room
```

>> BLOAD PIC.TB <<

by Rick Chapman

FUNCTION: BLOADs a Hi-Res image approximately three times faster than normal DOS 3.3.

LENGTH: 440 bytes ($1B8)

SYNTAX: &"NAME","filename",pagenumber
&"NAME","string",aexpr

SAMPLE: &"BLOAD PIC","PIC1",2

=> Loads the binary image named PIC1 into Hi-Res page 2.

HOW TO USE IT: This function is similar to the DOS command BLOAD filename,A$2000 when Hi-Res page 1 is specified or BLOAD filename,A$4000 when Hi-Res page 2 is specified. The differences are three-fold:

First, the format is obviously different. The filename is placed in quotes and the Hi-Res page is specified directly. Furthermore, the desired drive cannot be specified, this command always makes use of the last drive accessed.

Secondly, the command checks the length of the file before loading to insure that it is in fact a Hi-Res image file. It will only load files which consist of 33 or 34 sectors.

Finally, the command is 3 times faster than a normal DOS BLOAD. For those of you interested in such esoteric matters, the speed improvement is greater than that obtainable using 9 descending skewing. Optimum skewing for this command is in fact 3 descending, but only two extra disk revolutions are required for the standard 2 descending skew.

This command supports all the usual DOS error codes.

LIMITATIONS: The pagenumber variable must either be 1 or 2. This command is designed to speed up ordinary DOS 3.3. It is not needed, and in fact will not work with, the enhanced DOS provided on the Chart 'n Graph Toolbox diskette.

>> BSAVE PIC.TB <<

by Rick Chapman

FUNCTION: BSAVEs a Hi-Res image onto disk.

LENGTH: 209 bytes ($D1)

SYNTAX: &"NAME","filename",pagenumber
&"NAME","string",aexpr

SAMPLE: &"BLOAD PIC","PIC1",2

-> Saves Hi-Res page 2 as a binary file named "PIC1".

HOW TO USE IT: This function is identical to the DOS command BSAVE filename,A$2000,L$1FFB when Hi-Res page 1 is specified or BSAVE filename,A$4000,L$1FFB when Hi-Res page 2 is specified. This command is not faster than normal DOS. It is simply included as the inverse function to BLOAD PIC.TB.

The lengths are specified as $1FFB instead of the more common $2000 in order to save one sector on the disk. This does not in any way affect the quality of the Hi-Res image stored.

All standard DOS error codes are supported by this command.

LIMITATIONS: The page number parameter must either be 1 or 2. Like the BLOAD command, this will not speed up the enhanced DOS provided on the Chart 'n Graph Toolbox diskette. This DOS is already optimized for speed and does not require, nor will it work with, either the BLOAD PIC.TB or BSAVE PIC.TB commands.

>> WINDOW.TB <<

by Rick Chapman

FUNCTION: This command can store a rectangular area of the screen into an integer array and then later restore that area. This provides the basis for writing programs that use windowing or pull down menus or it can be used for simply moving image areas around on the screen.

LENGTH: 386 bytes ($182)

SYNTAX: &"WINDOW",dir,xbgn,xend,ymin,ymax,page,
intarray

&"WINDOW",aexpr,aexpr,aexpr,aexpr,aexpr,aexpr,
intarray

SAMPLE: &"WINDOW",1,0,3,151,180,1,X%(0)

=> Moves the contents of the specified rectangular area of Hi-Res page 1 into the integer array X% and then blacks out that area. The specified area is the first 4 bytes (0 through 3) of lines 151 through 180, that is $0 \leq X \leq 27$ and $151 \leq Y \leq 181$.

&"WINDOW",0,6,9,141,170,1,X%(0)

=> Moves the contents of the integer array X% back into an area of Hi-Res page 1 that is 42 pixels to the left and 10 pixels below the area that was stored in the instruction above.

HOW TO USE IT: This command will either move a rectangular area into an integer array or move the contents of that array back onto an area of the screen. The command is quite fast and was designed to make it easy to develop sophisticated looking graphic displays and menus.

All of the variables must be specified, none are optional. The direction variable determines the direction of the data flow according to the following table:

direction	action
0	array -> graph
1	graph -> array with clear
2	graph -> array without clear

A direction variable with a value of 1 moves the specified area into the array and then clears the specified area on the screen to black. This allows the user to fill in the blackened area with anything you might want, such as a menu of options, and then later restore the area to its original contents. A direction variable of 2 performs the same function but does not clear the area. This can be used if you want to make a copy of the area onto another part of the screen. A direction of 0 restores the contents of the area on the screen without disturbing the contents of the array.

The xbgn and xend variables literally specify the beginning and end bytes in each line of the area to be affected. Each byte contains 7 pixels so this command will only move areas in increments of 7 pixels wide. This was done to minimize memory usage and maximize speed, but turns out to be quite handy since the characters used in AMPERCHART are exactly 7 pixels wide (including the space around them). The xbgn and xend variables can range from 0 to 39 and xend must be greater than xbgn.

The ymin and ymax variables specify the lower and upper boundaries (in standard AMPERCHART coordinates) of the area to be affected, respectively. These variables must lie in the range 0 to 191 and ymin must be less than ymax.

The page variable specifies the Hi-Res page to be used by this command. A value of 1 or 2 specifies that the graphic area to be affected lies on that Hi-Res page. A value of 0 is also allowed and specifies that the area lies on the Hi-Res page that was last drawn to or used.

The array that is used by this command must be specified as an integer array having a single dimension. The subscript to the array must be included and should be set to zero. If an integer array of that name has already been dimensioned, then the command makes sure the array is large enough to hold all of the data being moved and then peforms the specified action. If the array is found to be too small, even if the direction of the transfer is from array to graph, a REDIM ERROR message is printed and the command will terminate. If no such integer array is found, one is created automatically. Thus you do not have to pre-dimension the arrays used by this command; it is in fact safer and simpler not to and let the command do all the work.

LIMITATIONS: The command only moves areas beginning and ending at byte boundaries. This command does not work with the Double Hi-Res commands included elsewhere on this disk. The following error messages are supported:

SYNTAX ERROR - The syntax of the command is incorrect.

ILLEGAL QUANTITY - One of the specified variables is out of range.

TYPE MISMATCH - The specified array has already been declared and is either not an integer array or has too many dimensions.

OUT OF MEMORY - There is not enough room in memory to create the specified array.

TECHNICAL NOTES: The command saves the data from the screen into the integer array by rows, proceeding from the left bottom corner of the area to the top right corner. If you want to use this command to get a portion of the screen into an array to manipulate, remember that the pixels on the screen are stored in reverse order in each byte, thus the least signficant bit of a screen byte is displayed to the left of bit 6 of that same byte. Also remember that bit 7 of each byte is not displayed, it is used only to determine color.

The integer arrays that are created by this command can be very large, up to 8K bytes. To figure the size of the array created, use the approximate formula: # of bytes in the array = (1+xbgn-xend)*(1+ymax-ymin)*2+7.

The command can also be used to move images or portions of images from page 1 to page 2 or vice-versa.

>> HI/LO ZOOM.TB <<

by Rick Chapman

FUNCTION: Will transfer a block of graphics information from either Hi-Res graphics page to the Lo-Res graphics page 1 (ZOOM). The inverse function mapping the Lo-Res graphics page into a block of the Hi-Res page (UNZOOM) is also supported.

LENGTH: 281 bytes ($119)

SYNTAX: &"NAME",x,y,zoomflag [,size]
&"NAME",aexpr,aexpr,aexpr [,aexpr]

SAMPLE: &"ZOOM",X,Y,0

=> Turns on the Lo-Res screen (page 1) and transfers a 40x48 box centered at Hi-Res location (X,Y) from the Hi-Res page to the Lo-Res page.

&"ZOOM",124,115,3,1

=> Turns on the Hi-Res screen, and transfers the top 40 lines of the Lo-Res graphics page to the box on the active Hi-Res graphics page that is centered about X=124 and Y=115.

HOW TO USE IT: The zoom takes a block of the active Hi-Res screen and maps it into the page 1 Lo-Res screen, pixel by pixel. The mapping is only black and white; color is ignored (colored points are treated as if they were white).

The geometry of the transfer is determined by the X,Y, and optional size variables. The nominal center of the Hi-Res box is specified by the X,Y coordinates. These coordinates must be within the normal permissable range for Hi-Res graphics (X in the range 0 to 279, Y in the range 0 to 191). The actual box is placed on the screen with its center as close as possible to these points, with the restriction that no portion of the box go outside of the screen. Thus if the X,Y coordinates are specified as (279,100) then the actual center of the box would be at (259,100).

The optional size variable sets the size of the box. There are two permissable sizes, one 40x48 that maps into the full Lo-Res screen (DEFAULT or size=0) and one 40x40 that maps into the top 40 lines of the Lo-Res screen (size>0). This latter mode provides for the use of mixed text and graphics on the Lo-Res screen.

The inverse operation, mapping the Lo-Res screen into a block on the Hi-Res screen is also allowed. The zoomflag selects between zoom (zoomflag=0) and unzoom (zoomflag>0). The geometry of the un-zoom is identical to that of the zoom.

The major difference between the operation of the two functions is that some limited color capabilities are available for the unzoom. The zoomflag can take on any value from 1 to 7 (0 is reserved for zoom), corresponding to the color that points to be plotted will appear on the Hi-Res screen. Points that are not plotted, corresponding to points that are blank on the Lo-Res screen, will be plotted in the color equal to (zoomflag-3). Thus if a black and white unzoom is desired, simply set zoomflag=3 for White1 and Black1 plotting, or zoomflag=7 for White2 and Black2 plotting.

EXAMPLES OF ZOOMFLAG

ZOOMFLAG	EFFECT
0	zoom
1	mixed color unzoom
2	mixed color unzoom
3	White1 plotting on Black1 background
4	mixed color unzoom
5	mixed color unzoom
6	mixed color unzoom
7	White2 plotting on Black2 Background

LIMITATIONS: The zoomflag must fall between 0 and 7. The size variable must fall between 0 and 255. Lo-Res page 1 is used for all zoom functions.

>> HI-RES EXPAND.TB <<

by Rick Chapman

FUNCTION: Will expand the size of any area on the active Hi-Res page by a factor of two.

LENGTH: 474 bytes ($1DA)

SYNTAX: &"NAME",xmin,xmax,ymin,ymax
&"NAME",aexpr,aexpr,aexpr,aexpr

SAMPLE: &"ZOOM",X-20,X+20,Y-30,Y+30

=> Expands the size of any image appearing in the box having a width of 40 and a height of 60 that is centered about X,Y. The expansion is by a factor of two, and the resultant image occupies the box having a width of 80 and a height of 120 that is centered about X,Y.

HOW TO USE IT: This function simply blows up a section of the active Hi-Res page. The section to be expanded is defined by the variables. The section which is filled, is twice the size in each dimension as the original and is centered on the original area. The Y-variables are given in standard AMPERCHART format (0 at the bottom of the screen, 191 at the top.)

The expansion is by a factor of two, so each pixel in the original area becomes a two-by-two set of pixels in the expanded image. Color is not preserved by this function; it should only be used on black and white images.

This function is most useful for creating oversize lettering. It is especially useful for creating large labels when used in conjunction with the MEDIAN FILTER. It can be used repetitively on the same area to achieve expansions of x4, x8, etc.

LIMITATIONS: The input variables must lie within the normal graphics range. In addition allowance must be made for the expansion by a factor of two. If the function is called with input variables that would lead to an expanded box that would leave the normal screen, an error message is returned and no action is taken.

This command is not very fast since it was designed to use a minimum amount of memory.

>> MEDIAN FILTER.TB <<

by Rick Chapman

FUNCTION: Implements a median filter over any specified area on the active Hi-Res page. This filter has the useful ability to smooth out the "blockiness" associated with the Hi-Res EXPAND command.

LENGTH: 446 bytes ($1BE)

SYNTAX: &"NAME",xmin,xmax,ymin,ymax,threshold
&"NAME",aexpr,aexpr,aexpr,aexpr,aexpr

SAMPLE: &"MF",X-20,X+20,Y-30,Y+30,4

=> Filters the box having a width of 40 and a height of 60 centered about X,Y with a threshold of 4.

HOW TO USE IT: This filter is capable of smoothing over the blockiness associated with the Hi-Res EXPAND command. It has been included in order to provide a memory efficient means of creating large character sets from the basic 5x7 characters used in AMPERCHART.

The box that the filter operates on is defined in the normal way by the X and Y variables. The threshold variable may range from 1 to 9. It defines the action of the filter.

The median filter acts by counting the number of pixels lit that are within a 3x3 box which moves all over the filtered area. If at any one location, the number of pixels lit in the 3x3 box exceeds the threshold, then the pixel corresponding to the center of the box in the filtered image will be turned on, otherwise that pixel will be turned off. The filter ignores color, so it should only be used in black and white applications.

The effect of the threshold value on the final result is best understood by seeing the results, so it is suggested that you RUN HI/RES ZOOM & MEDIAN FILTER DEMO on the AMPERCHART Diskette. This demo expands six sets of test characters on the screen, and then filters each of them with a different value for the threshold. Note the thresholds that yield useful results. In general the smaller the threshold, the fatter the character.

When filtering inverted labels (lettering in black, background in white) the effect of the threshold is also inverted. For inverted labels, larger thresholds yield fatter characters. To achieve the same net effect for filtering an inverted label as one would for filtering a regular label, use an inverted threshold of ten minus the regular threshold. If all of this is confusing, run the demo and experiment with it yourself. (Try inserting an &RVS instruction into the demo just after the image is first drawn.)

LIMITATIONS: The input variables must lie within the normal graphics range. As for all of the commands used in this package, the Y variable is reversed from that of standard Applesoft. Points at the top of the page have Y-values of 191 and points at the bottom have Y-values of 0.

This command has been designed to minimize the use of memory. This led to a compromise in the ability of the filter to operate and successfully deal with the edges of the area to be filtered. Thus, the area to be filtered is in general one pixel less in each direction than otherwise specified. Also the filter makes use of the left and upper edges of the box for temporary storage, so any pixels along these edges will be lost. For this reason it is best to have the edges of the filter box lie entirely on a solid white or black region.

>> THREE-D ARRAY.TB <<

by Rick Chapman

FUNCTION: Transforms a three-dimensional array into a two-dimensional projection.

LENGTH: 1146 bytes ($47A)

SYNTAX: &"NAME",ARRAY,A,B,G,SX,SY,SZ,XT,YT,ZT,ZO
&"NAME",array,aexpr,aexpr...,aexpr

SAMPLE: &"THREED",Q(0,0),A,B,G,SX,SY,SZ,XT,YT,ZT,ZO

=> Maps the array of x,y,z values in array Q into two dimensions through a projection defined by rotation angles A, B, and G, scaling variables SX, SY, and SZ, translations XT, YT, and ZT, and perspective ZO.

HOW TO USE IT: This function allows you to transform three-dimensional data into two-dimensional data to be plotted on the screen. This transformation is a true projection allowing you to draw a three-dimensional object as it would appear if you were looking at it at a particular angle and at a particular distance. The function requires a relatively large number of rather unusual variables, but don't let that worry you; it is quite easy to use.

The three-dimensional data to be transformed is passed to the command in a two-dimensional array. The array must be dimensioned to have 3 elements in its first dimension. The second dimension of the array can be of arbitrary length. Thus a legal array would be:

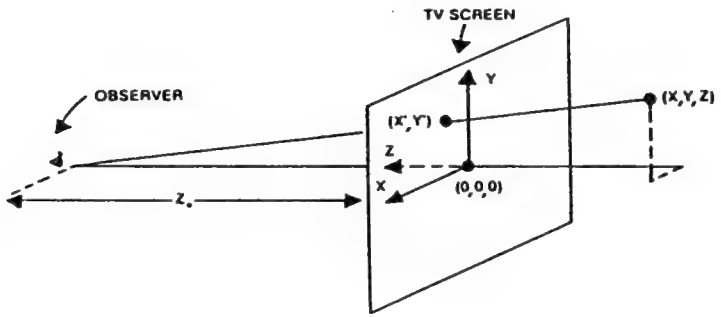
```
10 DIM Q(2,29)
```

Such an array would hold 30 (0-29) three-dimensional points. The X, Y and Z values for each point are contained in the Q(0,I), Q(1,I), and the Q(2,I) elements respectively.

The command modifies the X and Y values in the input array in order to return the projected X and Y points. Thus the one array serves as both input and output for the command.

The other variables specify how the points are to be rotated, moved, and scaled for the projection. To understand these variables let's first review our coordinate system.

The three-dimensional coordinate system is as shown in this figure.



The positive X-direction is to the left, the positive Y-direction is straight up, and the positive Z-direction is pointing out of the page (in the drawing shown as a diagonal). Note that there are other possible coordinate systems, but this is the one we will be using.

Now imagine a three-dimensional object which is defined by a set of points in our coordinate system. The transformation of that object begins by rotating it to some different orientation. First we rotate it A (for alpha) radians around the X axis. Then we rotate it B (for beta) radians around the Y axis. Finally we rotate the object G (for gamma) radians around the Z axis. This, believe it or not, allows for any rotational orientation of the object. When trying to use these rotations be sure to keep the order of the rotations in mind. No rotation is indicated by setting all three angles to zero.

Once the object has been rotated then it can be re-scaled along any or all of its dimensions. This allows you to expand or shrink the object along any of its dimensions by setting either SX, SY, or SZ to a number greater or less than one. Note that this re-scaling is performed after the object is rotated.

If you want to re-scale the object along a particular dimension before the rotation you should multiply the input array values by a fixed value before passing them to this command. No re-scaling is indicated by setting all three variables to one. If these variables are set to zero, the output values will all collapse into a single point.

The next step is to translate or move the object to any point in space. The object is moved by amounts given by the XT, YT, and ZT variables, each giving the distance that the object is moved along each particular axis. Thus to move the object to the left, XT should be set to a positive value. No translation is indicated by setting all three variables to zero.

The final step in the transformation is to project the rotated, scaled, and translated points onto a two-dimensional plane. The object is always projected onto the X-Y plane corresponding to Z=0. The ZO variable determines the center or viewpoint of the projection. This is the same type of projection that occurs in the eye, and the ZO variable can be thought of as where on the Z axis you want to place your eye.

All of these variables are required; there are no defaults. The command performs all of the necessary sine and cosine calculations only once for each array, so it is quite fast. The inclusion of the scaling variables does not significantly slow the command down if they are not used (by setting them to one) since Applesoft is very fast at multiplying by one.

An example of using this command is provided on the AMPERCHART diskette. THREE-D ARRAY DEMO shows the use of this command for plotting surfaces. Notice that the plot is made from the rear forward allowing for a particular simple minded hidden line algorithm (erase everything below the last drawn line). This hidden line algorithm only works for positively valued peaks but indicates a simple approach to a difficult problem.

LIMITATIONS: This command cannot be called from immediate mode; it can only be called from within a program. The algorithm used cannot handle points that after rotation, scaling, and translation have a Z coordinate equal to the perspective variable ZO. In these cases a CAN'T DIVIDE BY ZERO error will be generated. The easiest solution is to change the perspective variable ZO to a different value.

>> DUMP.*.TB <<

By Andy Scheck
and Rick Chapman

FUNCTION: This is a set of graphics dump commands that are capable of producing a hardcopy of the standard Hi-Res graphics currently on the Apple screen. (For Double Hi-Res dumps see Chapter 9). These commands are fully Toolbox system compatible and offer the advantage that they can be used from within any program. For hardware requirements see the installation section below.

LENGTH: (depends on command, see below)

SYNTAX: &"NAME"[,pn][,t]
&"NAME"[,aexpr][,aexpr]

SAMPLE: &"DMP",1,10

=> Dumps the contents of Hi-Res page 1 to the selected printer with a left margin of one inch (10 tenths of an inch)

&"DMP",3,0

=> Dumps the contents of Hi-Res page 1 and Hi-Res page 2 to the selected printer with no left margin. The two pages are printed next to each other so that double resolution plots can be generated.

AVAILABLE ROUTINES:

There are 9 available commands: three for the Epson MX-80, two each for the Epson MX-100, the Anadex DP9501/9001, and the Anadex DP9500/9000, and one for the Apple Imagewriter. (One of the commands works for both Epson printers.) Note that the MX-100 commands will also work with both of the new FX series printers from Epson.

Each of these commands requires a different set of variables and each serves a slightly different function which is designated by its name. The first part of the suffix to DUMP. is the printer type. The second part is the the figure size (1,2, or 3) and the last part designates the pages that are dumped (1-single page, 2-double page, or 12-both single and double):

NAME	PAGE NUMBER	TAB POSITION PN<3	TAB POSITION PN=3	SIZE BYTES
DUMP.MX80S1P1.TB	0-2	0-33	NA	236
DUMP.MX80S1P2.TB	NA	NA	NA	217
DUMP.MXS3P1.TB	0-2	NA	NA	276
DUMP.MX100S1P12.TB	0-3	0-90	0-43	301
DUMP.9500S1P12.TB	0-3	0-85	0-38	395
DUMP.9500S2P1.TB	0-2	0-26	NA	363
DUMP.9501S1P12.TB	0-3	0-94	0-57	405
DUMP.9501S2P1.TB	0-2	0-57	NA	375
DUMP.IMAGE.S1P12.TB	0-3	0-41	0-1	405

HOW TO USE IT: Each of these commands can provide a graphics dump on the appropriate printer and with the appropriate interface. Before you use any of these commands read the section below describing the hardware requirements and the print driver installation program.

The two possible variables for each command are the page number (PN) and the tab position (T). The permissible values for each command is shown in the table above. An entry in the table of 'NA' indicates that the variable is not applicable and therefore should not be entered. In general the permissible range for the page number variable is 0-3, indicating dump current Hi-Res page (0), page 1 (1), page 2 (2), or both pages side by side (3).

The tab position can be used in those commands which do not substantially fill the page. It allows you to position the graphics dump on the page. The values for the tab position are always given in tenths of an inch. The size 1 dumps are normally small, so that the tab position variable is available for them. The larger size 2 or 3 dumps have no position variable associated with them.

Note that no extra line feeds are generated before or after the screen is dumped (except when using the Imagewriter printer). This allows pictures to be vertically concatenated.

Each of the commands is described briefly below:

MX80S1P1 Single size, single page dump. Tab positioning allows you to left justify, center, or right justify on the page.

MX80S1P2 Single size, double page dump. Prints Hi-Res pages 1 and 2 side-by-side providing a means of obtaining double the resolution of a normal plot. This technique is discussed in Chapter 11. Only the leftmost 200 pixels of Hi-Res page 2 are actually printed.

MXS3P1 Triple size, single page dump. Will work with either the MX80 or the MX100. The increased size of the image prevents the use of tab positioning. This command makes use of fractional pin printing and thus requires a Graftrax Plus ROM in the printer being used.

MX100S1P12 Single size, single or double page dump. This is a combination of the first two commands set up especially for the MX100. The tab positioning is automatically limited to the proper values depending on the number of pages dumped. Since the MX100 is wider than the MX80, all of Hi-Res page 2 is printed.

9500S1P12 Single size, single or double page dump. Will work for either the DP9500 or the DP9000, with the restriction that printing should not be allowed to occur past 480 pixels when used with a DP9000. This means that double page printouts are not supported for the DP9000. (Attempts to use this feature may lead to the printer hanging up.)

9501S1P12 Single size, single or double page dump. Will work for either the DP9501 or the DP9001, with the restriction that printing should not be allowed to occur past 600 pixels when used with a DP9001. Double page printouts are supported for both printers, but you should be careful not to enter too large of a tab positioning variable when using a DP9001.

IMAGE.S1P12 Single size, single or double page dump. Only works with the Apple Super Serial Card Interface or an an Apple //c.

SETUP AND INSTALLATION

All of the dump commands, except for the Imagewriter command, come equipped with driver software compatible with either the Apple Parallel Card, the Epson Parallel Card, or the Tymac Parallel Card located in slot 1. These are the only non-intelligent parallel cards that we have actually tested. If, on the other hand, you have a different type of parallel card, try using the software as is; it will probably work.

If you have either a Grappler, an Apple serial interface, a CPS card, a Micro Buffer II, or a parallel card in a slot other than 1, then you must install a new printer driver using the program PRINTER CONFIGURE on the AMPERCHART Diskette. This program is menu driven, and will allow you to modify any or all of the dump commands (including the Double Hi-Res dump commands described in Chapter 10) to work with any of the above interfaces.

The Imagewriter dump command works only with the Apple Super Serial interface card (or the standard Apple //c serial interface). The command comes configured for the interface being in slot 2, but you can modify it to make use of any slot by running PRINTER CONFIGURE.

If you don't own one of the above interfaces and/or you own a different printer, then we recommend the purchase of PRINTOGRAPHER, also distributed by Roger Wagner Publishing, Inc. PRINTOGRAPHER will allow you to perform dumps with any combination of printer and interface and no technical or programming knowledge of your printer is required.

These dumps can be called from within your program, as is done in this package, or can be made from disk, as is done by most other packages. PRINTOGRAPHER also allows much greater flexibility in the final printout, is Routine Machine compatible and is reasonably priced. Write or call Roger Wagner Publishing for more information.

LIMITATIONS: None other than the limitations associated with the individual commands, and the hardware limitations listed above.

>> IDENTIFICATION.TB <<

by Rick Chapman

FUNCTION: Allows you to identify the configuration of the Apple your program is running on.

LENGTH: 297 bytes ($129)

SYNTAX: &"NAME",result
&"NAME",avar

SAMPLE: &"ID",RE

=> Returns the sum of the following codes in RE: (RE = R1 + R2)

R1	Apple Configuration
128	//e or //c with extended 80 column card (128K system)
64	//e with standard 80 column card (64K system)
32	//e but no Apple //e 80 column card (64K system)
0	Not an Apple //e (could be anything from an Apple][Plus to an IBM 3033)

R2	Apple // Family
0	Apple //e without Mouse Icon support
1	Apple //e with Mouse Icon support
2	Apple //c
3	Unknown member of Apple // family

HOW TO USE: Simply install and call the program. The return code will be passed back in the floating point variable specified. This command allows you to write programs that can reconfigure themselves or change their characteristics depending on what type of machine it is being run on.

Note that Mouse Icon support does not imply that a mouse is present on a particular //e. This simply means that the updated //e ROMs are being used. In addition, note that an Apple //e will always return a code of 130. The unknown member of the Apple // family result is included for possible future Apple expansions. It is not based on any currently known machine.

LIMITATIONS: This command cannot determine if an Apple //e with an extended 80 column card is in fact capable of displaying Double Hi-Res graphics. There is no way to distinguish a Revision A motherboard from a Revision B motherboard from within software and there is no way of telling whether a jumper has been placed across the molex connector on the 80 column card. If your program needs to determine this information, you will have to have it ask the user in order to be 100% sure. To add insult to injury, this command will fail to function correctly when run on an IBM 3033.

DEMONSTRATION PROGRAM: IDENTIFICATION DEMO

>> TONE.TB <<

by Craig Peterson

FUNCTION: Generates a pure tone of a given pitch and duration. Can also be used to pause.

LENGTH: 76 bytes ($4C)

SYNTAX: &"NAME" [,aexpr [, aexpr]] [;aexpr [, aexpr]]
&"NAME" [,pitch [, duration]]

SAMPLE:
```
&"TONE"
&"TONE",P
&"TONE",P,D
&"TONE",P+5,D/2;P2,D2;P3,D3;P4;P5;P6
```

HOW TO USE IT: The TONE.TB command can be used in a variety of ways. If no variables are included after the name, then a tone having about the same pitch and duration of the normal Apple beep will be sounded.

If one variable is included after the command name, it will change the pitch of the tone. This variable can be any numeric constant, variable or expression, but must have a value from 0 to 255. Lower values cause higher pitches. A value of '0' produces no sound. If the pitch value is doubled, then a tone about one octave lower will be produced.

If a second variable is given, the length of the tone can be changed. The length variable can also be any numeric constant, variable or expression having a value between 0 and 255. This length value is determined in approximately 1/100's of a second, so '100' would produce a tone of about one second in length. This allows tones with lengths ranging from 1/100 to over 2.5 seconds in length.

With pitch set to '0', the duration can be adjusted to produce silent waits of 1/100 to 2.5 seconds.

In addition, multiple tone variables can be included in a single command statement by using a semi-colon to separate each new TONE command. Variables following each semi-colon can either be a pair to specify both pitch and duration, or just a single pitch variable.

Be sure to run the TONE DEMO to see what the musical possibilities of this command are.

LIMITATIONS: Both pitch and duration values must be numeric variables in the range of 0-255.

SAMPLE LISTING:

```
 1 CALL PEEK(175) + 256 * PEEK(176) - 46
10 INPUT "PITCH, DURATION?";P,D
20 &"TONE",P,D
30 GOTO 10
```

Here is a table of the note values produced with different values for pitch:

NOTE	PITCH NO.	PITCH NO.	PITCH NO.
F SHARP	135	67	33
F	143	71	35
E	151	75	37
E FLAT	160	80	40
D	170	85	42
C SHARP	180	90	45
C	191	95	47
B	202	101	50
B FLAT	214	107	53
A	227	113	56
A FLAT	241	120	60
G	255	127	63

AND.. 31 (G)

DEMONSTRATON PROGRAM: TONE DEMO

>> RESTORE AMPERSAND.TB <<

by Craig Peterson

FUNCTION: This will restore the ampersand vector to its original value, as it was before the Applesoft program was run.

LENGTH: 18 bytes ($12)

SYNTAX: &"NAME"

SAMPLE: &"AMP"

HOW TO USE IT: If you are using any utilities which use the ampersand vector, running one of your own programs with Toolbox commands in it will re-write that vector. When your program ends, the previously operational utility (if there was one) will no longer respond to the ampersand in the immediate mode.

To correct this, an optional procedure has been provided by way of this command. When your program first connects the ampersand vector via the hook-up on line #1, the Interface Routine stores the original value of the ampersand vector.

When you are going to END the Applesoft program, simply invoke the RESTORE AMPERSAND.TB command and the ampersand will be restored to its earlier function.

LIMITATIONS: None per se, although a fair degree of care is required in its use. If you exit a program via an error, control-C, RESET or any other manner other than the controlled (and expected) exit you previously set up, it is likely the RESTORE AMPERSAND.TB command will not be called. This would then leave the ampersand still pointing to the Interface Routine. If you were then to re-run your program, the Interface Routine would again save the current status of the ampersand, which would now point to itself. In other words, any previous ampersand related utilities would now be permanently disconnected.

NOTE: If you think you understand how the command RESTORE AMPERSAND.TB is meant to be used, then it is good practice to use it in all programs containing Toolbox commands. It MUST, however, be the LAST command used before your program ceases execution, since the use of this command disables ALL use of any further Toolbox commands.

If you use RESTORE AMPERSAND.TB, and then try to use another Toolbox command, the result will depend on the pre-RUN value of the ampersand vector, but a likely result would be a SYNTAX ERROR.

SAMPLE LISTING #1:

```
  1 CALL PEEK(175) + 256 * PEEK(176) - 46

 10 REM USUAL APPLESOFT PROGRAM HERE

100 &"AMP":END
```

You can also use RESTORE AMPERSAND.TB to "toggle", or switch back and forth, between the Toolbox ampersand commands and other ampersand-driven commands that your program may be using.

The key to the system is to remember that the CALL PEEK(175) + 256 * PEEK(176) - 46 statement hooks up the Toolbox commands, and the RESTORE AMPERSAND.TB command reconnects whatever previous ampersand commands (if any) existed when the CALL was done.

The technique, then, is to use the CALL statement to turn on Toolbox commands, and the RESTORE AMPERSAND.TB command to turn on your own ampersand commands. The possible approaches are illustrated in listing #2.

SAMPLE LISTING #2:

```
  1 PRINT CHR$(4);"BRUN AMPERSAND.USER.PROGRAM"
  2 REM INSTALL YOUR OWN AMPERSAND UTILITY HERE
 10 CALL PEEK(175) + 256 * PEEK(176) - 46
 11 TOOLBOX SYSTEM REMEMBERS IT
 10 REM A TOOLBOX COMMAND HERE
 20 &"AMP": REM SWITCH TO USER AMPERSAND FUNCTION
 30 &X,Y,Z : REM A USER AMPERSAND FUNCTION
 40 CALL PEEK(175) + 256 * PEEK(176) - 46
 41 REM RE-INSTALL TOOLBOX COMMANDS
 50 &"AMP": REM SWITCH TO BACK TO USER AMPERSAND
 60 &A,B,C : REM ANOTHER USER AMPERSAND FUNCTION
100 END
101 REM IN THIS SETUP, THE AMPERSAND CANNOT BE
       RETURNED TO THE NORMAL APPLESOFT CONDITION
       WHEN YOUR PROGRAM IS OVER, BECAUSE THE
       USER AMPERSAND FUNCTION CANNOT RESTORE IT
       ITSELF, AND AMPERSAND RESTORE CAN ONLY
       SWITCH BACK TO THE USER AMPERSAND FUNCTION.
```

5.15 AMPERCHART Alternative Shape Tables

AMPERCHART's character set is created using shape tables which contain definitions of all the letters in the alphabet.

Alternate shape tables can be useful however, and included in this package are a set of commands that support the use of alternate shape tables with AMPERCHART. A configuration program allows you to generate an AMPERCHART shape file, using your shape table, which when executed links the shape table automatically to AMPERCHART. No special knowledge is required! In addition, three preconfigured AMPERCHART shape files are included on this diskette: a full upper and lower case ASCII set, a double size character set for large labels and a set of graphic icons.

This section describes the general techniques used to implement the new shape tables. Later sections discuss the pre-configured shape tables in more detail.

5.15.1 Using the New Shape Tables

There are two general steps required to use a new shape table with AMPERCHART. First, an AMPERCHART shape table must be created by attaching your shape table to a special Toolbox compatible table. This need only be done once for each shape table. Secondly, you must use the Workbench to add the new shape table to your program, along with the usual files such as RELOCATE I.TB and AMPERCHART JUMP FILE.TB.

When your program is run, you determine the location of AMPERCHART in the computer (usually using the LOC command) and then pass this address to the new shape table. This completes the setup; AMPERCHART will now use the new shape table for all labeling. A second call to the shape module will reinstall the standard AMPERCHART characters.

With this overview, we can discuss the individual steps in detail. The next subsection describes how to create a shape module. If you just want to use those modules that are already included on this diskette then you can skip the next subsection and proceed directly to subsection 5.15.3.

5.15.2 Creating an AMPERCHART Shape Table

A shape module is created by running the CONFIGURE AMPERCHART SHAPE MOD program on the back of the AMPERCHART diskette. The program will print a hello page and then load into memory the core shape module: AMPERCHART SHAPE MODULE CORE. This module is a stripped version of the final shape module and therefore you should not attempt to execute it directly.

The program will then ask you to insert the disk containing the shape table you want to use. The shape table will then be loaded into memory.

The program will proceed by asking for the number of the shape that you want to correspond to the letter 'A'. The shapes in the shape table are numbered from 1 to N, where N is the total number of shapes in the table. When AMPERCHART accesses the shape table from a HLB or VLB command, it automatically translates the input character string into a sequence of shape numbers. By answering this question, you are telling AMPERCHART which shape it should print when it would normally be printing the letter 'A'. For "standard" character shape tables that begin with the '!" character, the usual value to enter here is '17'. All of the rest of the shapes will be printed in corresponding order in response to an ASCII input character.

After entering this value, the program will set up the shape module in memory, and then ask for the name that you want the program to save the new shape module under. The program will automatically add '.TB' to the end of the name (unless you have already included .TB in the name) and save the new shape module out to disk. You can now use the new shape module with AMPERCHART.

5.15.3 Using AMPERCHART Shape Tables

AMPERCHART shape tables are Toolbox files with a particular shape table in them. They are only useful in conjunction with AMPERCHART. They cannot be used to install shape tables into the machine without AMPERCHART (in order to do this, see the SHAPE GOBBLER program on the Wizard's Toolbox package).

To use a shape table, both AMPERCHART and the shape table must be installed somewhere in the machine. The type of installation and the location are unimportant. The syntax for telling AMPERCHART to use the new shapes

when both AMPERCHART and the shape table are attached using the Workbench is:

```
125 &"CHART",LOC(ADR) : &"NEW SHAPE",ADR
```

Alternatively, if AMPERCHART is installed in a fixed location (see Chapter 7) with the ampersand vector pointing at it and the shape table is installed in different fixed location, say at address 24576, then the calling syntax would be:

```
125 &LOC(ADR) : CALL(24576)ADR
```

The possibilities are endless, but the basic structure remains the same: first you use the LOC command in AMPERCHART to find out where AMPERCHART is located (for novice users, this is one of the few times you'll need this command). Then you call the shape table, passing the address of AMPERCHART to it as a variable.

The shape table will then rewrite a small section of AMPERCHART, setting locations within the version of AMPERCHART currently in memory to point to the shape table. The changes are only made to the version of AMPERCHART in memory, the copies on disk are unaffected. After installation, any labeling done in AMPERCHART will use the new character set.

If you want to revert back to the standard character set, even temporarily, then simply call the shape table a second time using the same syntax as before. This will reinstall the standard pointers in AMPERCHART. A third call will reinstall the new character set, etc. Multiple shape tables can be used by installing several shape tables into the machine, but only one can be used at a given time.

The shape tables will work with all currently released versions of AMPERCHART or DHR AMPERCHART.

5.15.4 Shape Table Requirements

The technique described above can be used to connect any shape table to AMPERCHART. Despite this, some shape tables are more convenient to use than others because of the structure of AMPERCHART. AMPERCHART assumes that the shape table it is using consists of a set of characters that fit into a 5-dot wide by 7-dot high grid and printed with an extra column of blank dots on either side. This assumption is used internally

whenever AMPERCHART computes the position to place a label and when AMPERCHART determines whether to clip a character or not. Thus shape tables consisting of larger characters can lead to some undesirable effects, such as overlapping of adjacent shapes and printing outside of the clipping window.

The upshot of all of this is that if you use a shape table based on anything other than a 5x7 matrix, then you will be responsible for the positioning of each particular character. There is, as in all things, a notable exception. Double size characters can be used without too much difficulty by double spacing the input string. This is described in more detail in the discussion of the double size character set that follows.

When using 5x7 size shapes, the starting point for each shape should be in the lower left corner of the grid. If it isn't, the position parameter will not behave correctly and clipping may be slightly in error.

Additionally, the first shape in any table is the default shape - the shape displayed when the HLB or VLB commands encounter an input character outside of the defined range.

>> UC & LC SHAPE MODULE.TB <<

by Rick Chapman

FUNCTION: This command links a new 5x7 character set that includes all of the printable ASCII characters, including lower case, into the currently installed version of AMPERCHART.

LENGTH: 1404 bytes ($57C)

SYNTAX: &"UC & LC SHAPE",ADR

SAMPLE: &"AMPERCHART",LOC(ADR) : &"UC & LC SHAPE",ADR

=> Installs the new full 5x7 shape table in the version of AMPERCHART beginning at ADR. A second call with the same syntax restores the standard 5x7 character set.

HOW TO USE IT: The command was developed to allow you to easily use both upper and lower case characters from within the confines of AMPERCHART. The shape table is included in a special Toolbox file designed specifically to link the shape table with whatever version of AMPERCHART you are using.

To use the command simply use the following instruction sequence:

```
1250  &"AMPERCHART",LOC(ADR) : &"UC & LC SHAPE",ADR
```

Note that the command requires that one version of AMPERCHART already be installed in the machine. After you are done with the optional characters, a second call of the same format will restore the original character set.

Once the full character set is installed into AMPERCHART, the HLB and VLB commands will use them instead of the standard characters. All of the clipping and position variables work as for the standard character set (except for the descenders that are used for lower case characters). To input the lower case characters on an Apple][Plus, you will have to use the CHR$ function to generate the correct ASCII codes.

LIMITATIONS: None

DEMONSTRATION PROGRAM: NEW SHAPES DEMO

>> S11X14 SHAPE MODULE.TB <<

by Rick Chapman and Andy Scheck

FUNCTION: This command links a double size set of alphanumeric characters into the currently installed version of AMPERCHART providing for the easy generation of large titles and labels.

LENGTH: 2217 bytes ($8A9)

SYNTAX: &"S11X14 SHAPE",ADR

SAMPLE: &"CHART",LOC(ADR) : &"S11X14 SHAPE", ADR

=> Installs a double size shape table in the version of AMPERCHART beginning at ADR. A second call with the same syntax restores the standard 5x7 character set.

HOW TO USE IT: The command was developed to allow you to easily use large size characters without having to blow them up and smooth them using other commands. The shape table is included in a special Toolbox file designed specifically to link the shape table with whatever version of AMPERCHART you are using.

To use the command simply use the following instruction sequence:

```
1250 &"AMPERCHART",LOC(ADR) : &"S11X14",ADR
```

Note that the command requires that one version of AMPERCHART already be installed in the machine. After you are done with the large characters, a second call of the same format will restore the original character set.

Once the large characters are installed into AMPERCHART, the HLB and VLB commands will use them instead of the smaller standard characters. When using these commands you must use double spacing in the input string to prevent overlapping characters [e.g. to print the string "TEST" at location 100,100 you would use: &"AMPERCHART",HLB("T E S T",100,100,-3)].

The other difficulties with using these characters is that the larger characters are not properly justified or clipped. The improper justification means that the position variable does not act as it does for the standard characters. In fact the only position variable value that works as it previously did is '3'.

With double spacing of the input string, the other values of position variable are nearly correct in the left-right sense, but in general they will be off several pixels in the up-down sense. The lack of proper clipping leads us to suggest that you always use a negative position variable to turn off the clipping effects.

Despite these problems, this command does provide you with a simple and fast means of creating large characters on the Hi-Res or Double Hi-Res screen. In fact, the Double Hi-Res screen allows up to 40 of these characters to be displayed on the screen in a single line (12 lines of 40 characters each).

LIMITATIONS: The large characters are not properly justified according to the position variable. In addition, the clipping functions of AMPERCHART do not correctly handle the new large characters so the clipping function should be turned off (negative position variable) when using the large characters.

DEMONSTRATION PROGRAMS: CALENDAR, BIG CHARACTERS

```
 !"#$%&'()*+,-./0123456789:;<=>?@
ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`
abcdefghijklmnopqrstuvwxyz{|}~
```

UC & LC SHAPE.RM allows you to use both upper and lower case characters from AMPERCHART with the HLB & VLB commands.

The Apple icons can also be used from AMPERCHART with the HLB & VLB commands.

>> APPLE ICONS MODULE.TB <<

by Rick Chapman

FUNCTION: This command links a set of 32 graphic icons, identical to those on the updated Apple //e, into the currently installed version of AMPERCHART.

LENGTH: 764 bytes ($2FC)

SYNTAX: &"APPLE ICONS",ADR

SAMPLE: &"AMPERCHART",LOC(ADR) : &"APPLE ICONS",ADR

=> Installs the 32 graphic icons in the version of AMPERCHART beginning at ADR. A second call with the same syntax restores the standard 5x7 character set.

HOW TO USE IT: The command was developed to allow you to use the graphic icons that are available on newer Apple //e's. The command operates exactly the same as the last two commands.

To use the command simply use the following instruction sequence:

```
1250  &"AMPERCHART",LOC(ADR) : &"APPLE ICONS",ADR
```

After you are done with the optional characters, a second call of the same format will restore the original character set.

Once the icons are installed into AMPERCHART, the HLB and VLB commands will use them instead of the standard characters. These icons are slightly larger than the standard character set (by one dot in width, two dots in height) so that the position and clipping are nearly correct. The icons map into the 32 ASCII characters starting at @ (@,A,B,C,...).

LIMITATIONS:None

DEMO PROGRAM: NEW SHAPES DEMO

6. DEMONSTRATION PROGRAMS - NOTES AND HINTS

There are a number of demonstration programs on the AMPERCHART Diskette, each illustrating a different set of AMPERCHART capabilities. These programs are individually described below:

LABEL TEST: Shows off the full plotting and labeling capabilities of AMPERCHART. Notice the use of the STU command at the beginning of the program to properly locate a label on the screen.

BAR CHART: Gives an example of double width characters and simulated 3d bars. A nice example to base your own bar charts on.

PIE CHART: A pie chart showing the utility of pie charts.

FINANCIAL: Another bar chart program, using colored bars, a grid, and inverse to highlight labels.

CHAPMAN PLOT: U.S. surplus and deficits for the 20th Century plotted on an interesting logarithmic scale named for its modest inventor.

THREE-D PLOT: A demonstration proving that all is not flat.

POLAR PLOT: Example of polar plotting, a pleasant change from all of these square plots. Also doubles as a sensitivity curve for your FM dipole antenna.

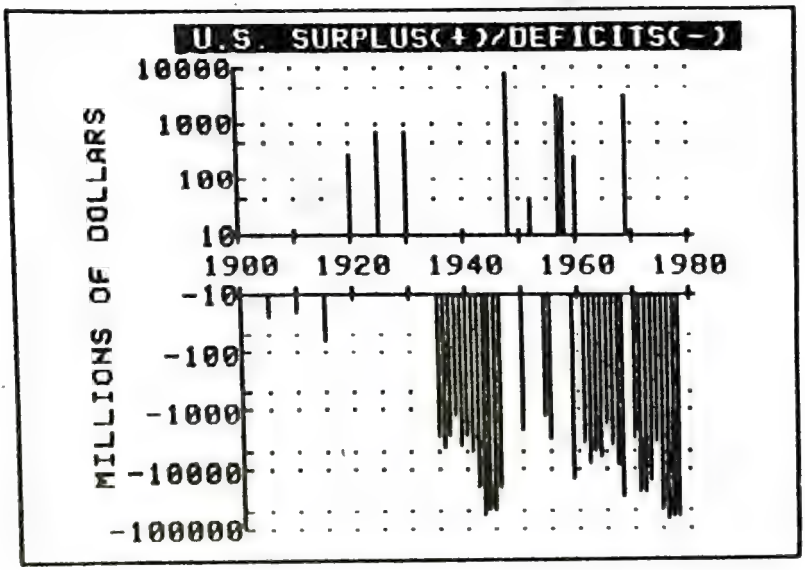
LOG CIRCLES: Plots circles using all the combinations of linear and logarithmic axes. Demonstrates the use of multiple clipping windows, the ARC & AXS commands, and the use of logarithmic scales.

STARS: Unmitigated fluff showing off the ARC command.

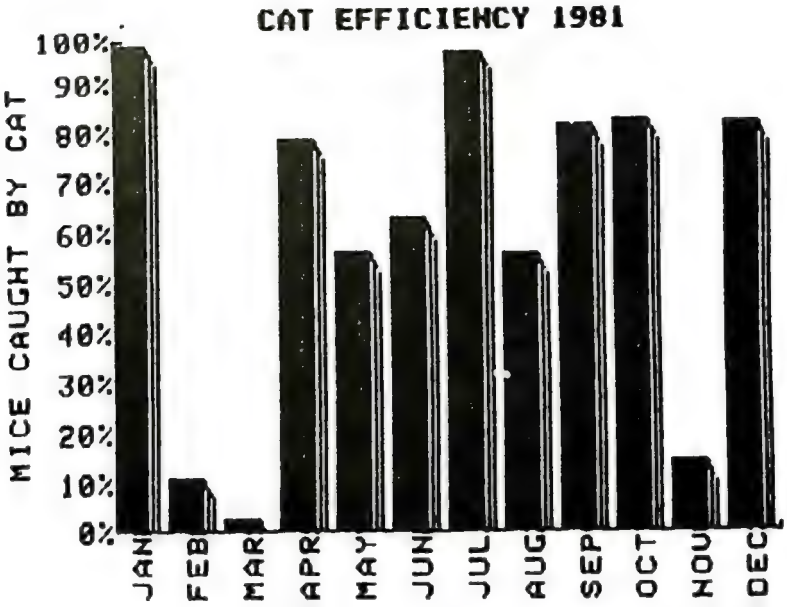
CHECKER BOARD: Uses the FIL & ARC commands. Notice the direction of the fill algorithm, first up, then down. This has no real significance, but it's nice to watch.

MIND SCRAMBLER: An interesting demonstration of the special effects possible using AMPERCHART. Check out how short this program is.

Chapman Plot



Bar Chart



POLYGONS: Some more ARC fluff.

POSITION: Demonstrates the use of the position parameter in horizontal and vertical labeling.

CHARACTERS: Displays the Hi-Res characters available for labeling. Notice how the responsive flashing cursor is implemented.

HIDDEN LINE DEMO: Just to show that hidden lines can be removed. The program can be used for other data by modifying the beginning portion of the program.

LITTLE ANIMATION: Uses two screens and the WRK and DSP commands to rotate a star.

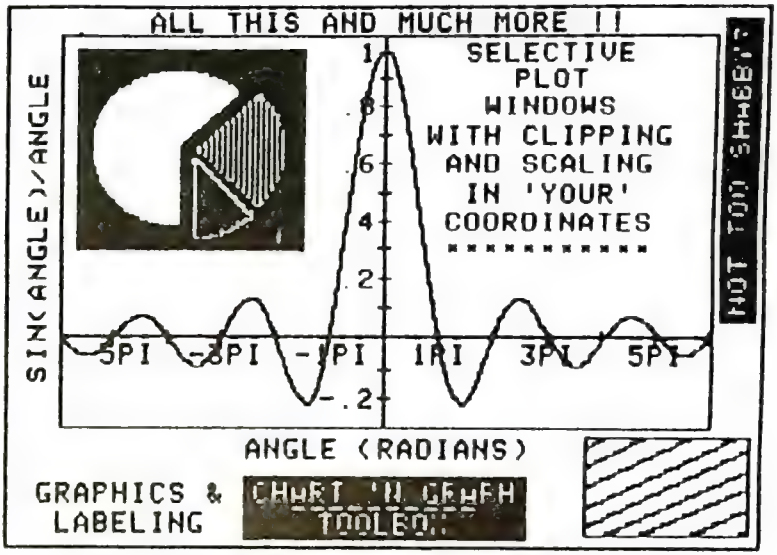
SAMPLE FLOW CHART: A different type of use for AMPER-CHART. Both screens are drawn on (don't blink or you'll miss the first). The space bar toggles back and forth between them; any other key will exit the program.

DOUBLE RESOLUTION PLOT DEMO: Demonstrates the method of generating super hi-res plots that's described in the Advanced Usage section. The use of printer dump routine is also demonstrated within the program, but the dump call has been placed within a REM statement so that the program can be run by any user.

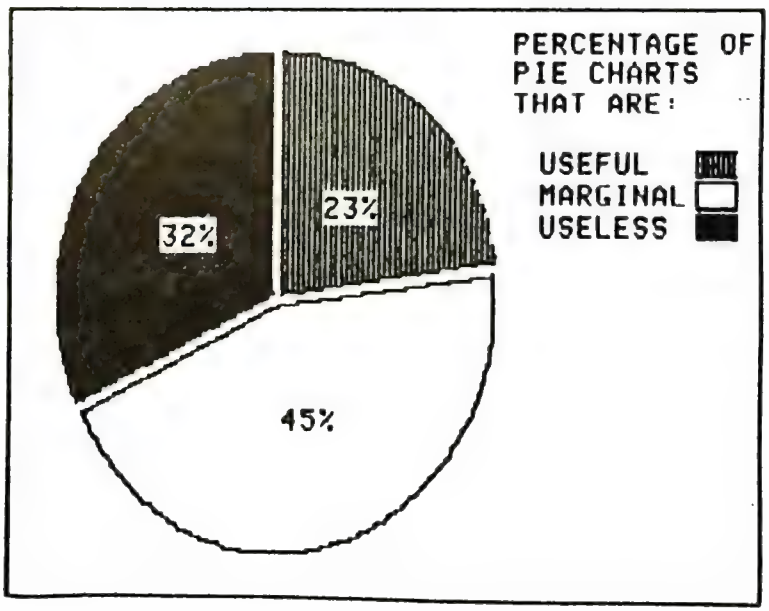
CONTOUR: The program generates a contour representation of a three dimensional surface using a simple contouring algorithm. The program is exceedingly slow, but can be useful for certain types of presentations.

BASIC GRAPHICS DUMP: An example of using the BIT command from Applesoft to print a graphics dump. If you don't own an Epson, an Anadex, or Printographer this example illustrates how you might be able to write your own dump routine. The program is currently set up for an Epson MX-80 with a Tymac parallel card.

Label Test



Pie Chart



OTHER DEMOS

SPLITTER DEMO: This program is not in fact a graphics program, but is a long program that will illustrate the vagaries of splitting, editing, and unsplitting.

HI/LO ZOOM DEMO: Plots the figure from LABEL TEST to the screen, then zooms it into Lo-Res, allowing you to pan around the picture.

HI-RES ZOOM/MEDIAN FILTER DEMO: First, the program demonstrates a zoom by 2, then a zoom by 4. Then the program demonstrates the effect of the median filter for six different values of the filter parameter.

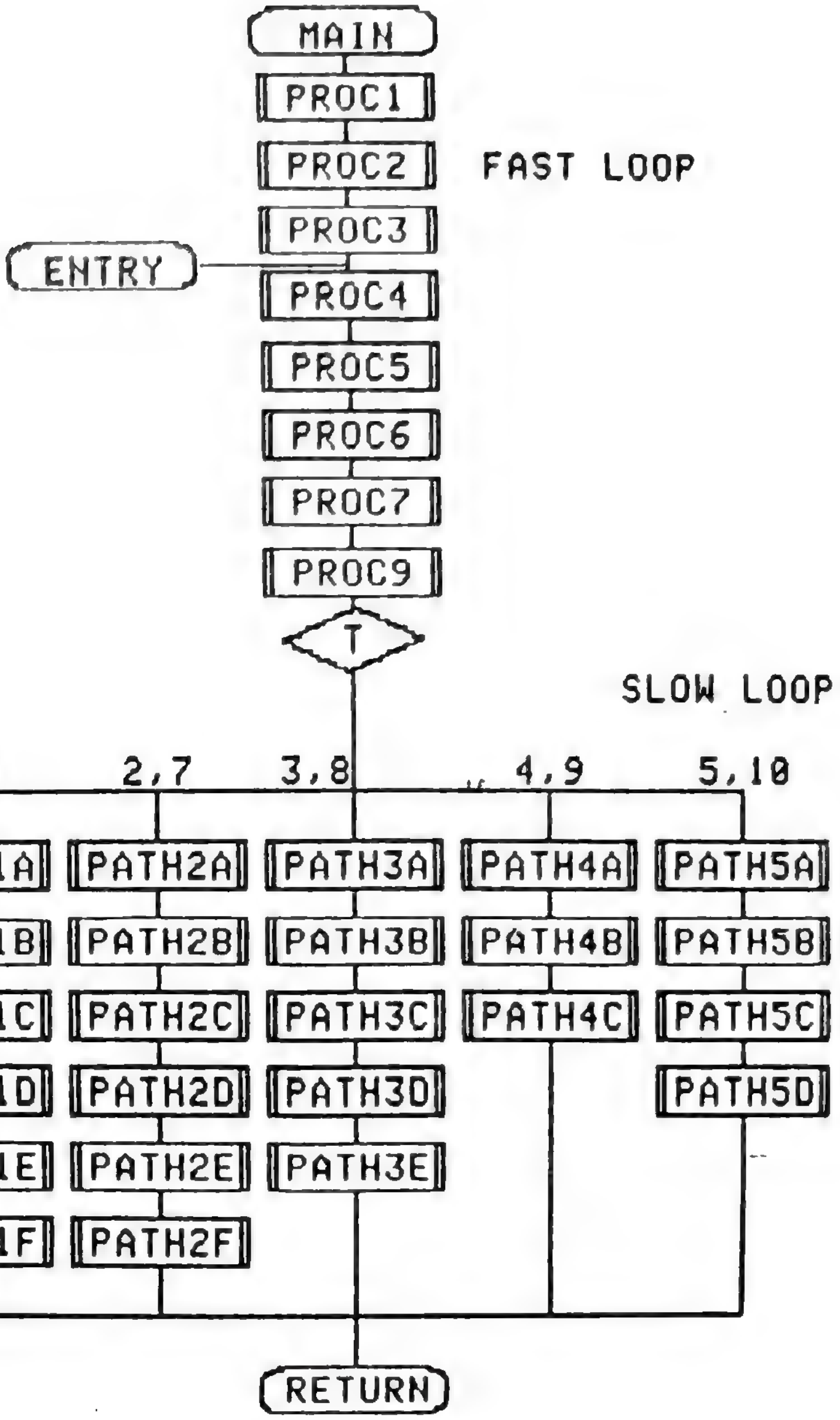
WINDOW DEMO: A simple minded demo showing off the capabilities of the WINDOW.TB module. The pull down menus generated in this program don't do anything, but they should give you an idea of how to use this routine in your own programs. (The Authors and RWP will provide free AMPERCHART T-shirts to the first one hundred users who write a complete LISA emulator for the Apple][Plus using this routine!)

NEW SHAPES DEMO: Demonstrates the use of alternative shape tables with AMPERCHART. A set of routines, described in Section 5.15 allow you to use any shape tables with AMPERCHART. This program demonstrates two of the three alternative character sets that come on this disk, a full ASCII character set with upper and lower case and a set of graphic icons identical with those being used in the new Apple //e and the //c.

CALENDAR: Demonstrates the use of double size characters with AMPERCHART (not to be confused with Double Hi-Res) using the S11X14 SHAPE.TB module. The routine only draws the calendar for the month of December 1983. If you want current calendars, you can always expand this program.

BIG CHARACTERS: Another demonstration of the large size characters. This program simply displays all of the large characters that are available using S11X14 SHAPE.TB and AMPERCHART.

Sample Flow Chart



7. AMPERCHART SYSTEM USAGE

The recommended way to use AMPERCHART.TB is by the indirect method of adding either RELOCATE I.TB or RELOCATE II.TB and AMPERCHART JUMP FILE.TB to your program using the Workbench utility.

The choice as to whether to use RELOCATE I.TB or RELOCATE II.TB depends on whether you intend to use page 2 of the Hi-Res display, such as with a GCO(2) or DSP(2) command.

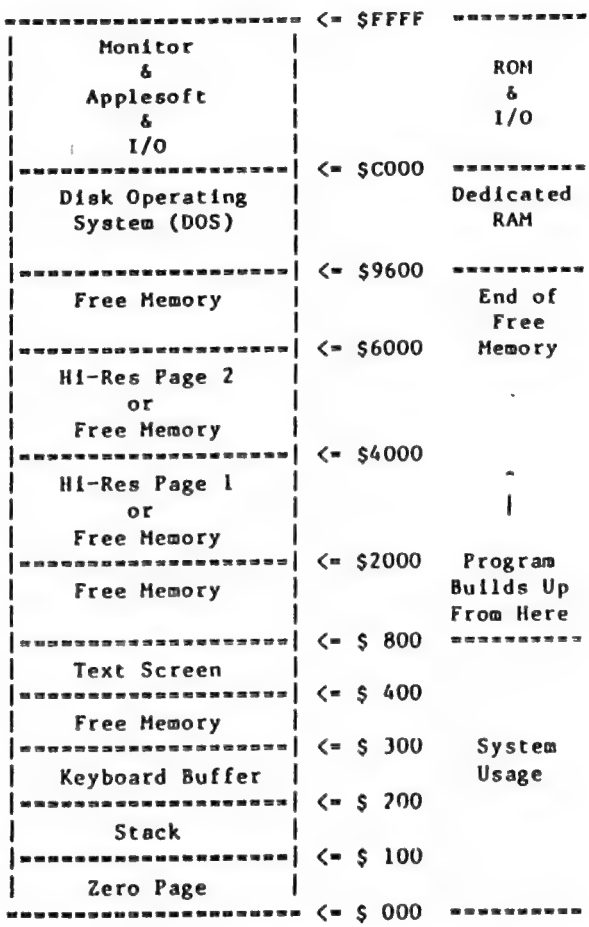
There are alternate methods of using AMPERCHART and the other Hi-Res commands included in this package, but before exploring them, a look at memory organization in the Apple would be helpful. The material in this section is not required for normal use of AMPERCHART and is provided only for your interest, and as possible alternatives should you desire them.

7.1 Memory Usage in the Apple II

The more advanced user will probably want to know more about the two previously described installation options. In addition to these, there are two more memory efficient installation options which can be used. Before describing all of these options in more detail, a brief description of memory usage within the Apple II is in order.

The 6502 microprocessor within the Apple II is capable of directly addressing 64K bytes (K = 1024) of information. Each byte can be thought of as a location, with some unique address, that contains a number from 0 to 255 ($00 to $FF, where '$' indicates hexadecimal). In hexadecimal, these addresses run from $0000 to $FFFF. (We will use hexadecimal notation throughout the rest of this manual, but don't worry if you aren't familiar with it; only the general overview is necessary.)

Within this 64K of memory space the following organization is maintained:



It is important to note that the memory available for Applesoft programs resides from $800 to $9600, and that Hi-Res pages are rather inconveniently located in the middle of this range. There are a number of ways to allocate this memory to AMPERCHART and your Applesoft programs. Several of these are discussed here.

7.2 Standard Toolbox Method

The most complete version of AMPERCHART is just slightly less than 6K in length. This matches quite nicely the 6K of free memory below the Hi-Res pages from $800 to $2000. When you use RELOCATE I/II.TB, the command makes use of this fact by installing AMPERCHART.TB below the Hi-Res pages and moving the location that the Applesoft program starts at to $4000 or $6000, depending on whether one or both Hi-Res pages will be used. This is the most efficient method in terms of its usage of memory, as very little memory is wasted. Using this method, programs of up to 21.5K bytes in length can be written and used in conjunction with AMPERCHART.

The Toolbox files RELOCATE I.TB, RELOCATE II.TB, and AMPERCHART JUMP FILE.TB are used to implement this mode of operation. When called, RELOCATE moves any Applesoft program that is currently in memory above either Hi-Res page 1 (RELOCATE I) or both Hi-Res pages (RELOCATE II) and then loads and installs AMPERCHART.TB below Hi-Res page 1. If AMPERCHART.TB has already been loaded, then no repeat load is done. Similarly, if the program has already been moved, then the command recognizes this and returns without further action.

When RELOCATE is called, it also checks to see if the ampersand character is active, i.e. pointing to within the program "envelope" (part of memory being used by the current Applesoft program). If so, the ampersand is left alone. If however, the vector is not currently active, the vector is set to point directly to AMPERCHART.TB.

This means that by typing in BRUN RELOCATE I.TB in the immediate mode, you can create a setup of AMPERCHART which does not use the Toolbox system. This is described in detail in section 7.5 later in this chapter.

After the RELOCATE command is used, the pointers will remain in effect until either the system is turned off, a new disk is booted, or the FP command is given.

If you want the system restored to normal when your program ends, there are two options, depending on whether the program in question is going to leave the user in the immediate mode (Applesoft prompt:]), or RUN another program.

If you simply want your program to end with the normal Applesoft memory setup restored, just use the DOS command "FP" as the last statement executed. For example:

```
99 PRINT"GOODBYE...": PRINT CHR$(4);"FP"
```

IMPORTANT: The FP statement will also erase the program in memory, as well as restore the normal memory setup. Be careful that you only use this exit after the program has been debugged and is in a finished stage. The main reason for this warning is that is easy to make a lot of changes to a program while you're working on it and then in testing the changes, execute the 'FP' exit which then erases the changes.

If your program is going to RUN another program when it is finished, as in the HELLO program which then runs EASY CHART, SHAPE MAKER and other programs, then a few simple POKEs will restore the system:

```
90 PRINT"GOODBYE..."
95 POKE 103,1:POKE 104,8
99 PRINT CHR$(4);"RUN PROGRAM #2":END
```

If you have converted your program to ProDos, or for some other reason don't want to use the FP method, the two previous methods can be combined as in this program:

```
90 PRINT"GOODBYE..."
95 POKE 103,1:POKE 104,8
99 NEW
```

The NEW statement still erases your program as in the 'FP' method, but does not rely on DOS. The only disadvantage to the NEW method is that it takes a few extra POKEs over the FP method, and FP does do a better job of resetting HIMEM and other internals of Applesoft.

One further note for the advanced user. The RELOCATE modules will work with some but not all versions of Ramcard DOS. The only way to determine if it works with a particular version of relocated DOS is to try it.

7.3 Using SPLITTER.TB and AMPERCHART.TB Directly

With the standard method, AMPERCHART.TB is loaded indirectly by RELOCATE, which at the same time rearranges memory to avoid conflicts with the Hi-Res pages.

SPLITTER is a special utility provided on the disk that also performs the function of avoiding conflict with the Hi-Res pages, but it does it a little differently. SPLITTER works by actually splitting the Applesoft program around the Hi-Res pages. (The program SPLITTER DEMO on the diskette gives a graphic illustration of this process.)

If for some reason, you do not like the necessity of maintaining AMPERCHART.TB as a file separate from your main program on a diskette, it can be directly added to your program like any other Toolbox command. In this case, neither RELOCATE or AMPERCHART JUMP FILE.TB whould be added and AMPERCHART.TB would be given the command name "CHART" (or whatever you like to use) directly. Because a program with AMPERCHART added to it is guaranteed to conflict with the Hi-Res pages, you also then need to add SPLITTER.TB.

When run, you then would call the SPLITTER command much the same way you would the RELOCATE command, i.e., at the very beginning of your program before any variables are defined. See the documention on the SPLITTER command for specific details on the options for this command.

A sample program using this approach might look something like this:

```
 1  CALL PEEK(175) + 256 * PEEK(176) - 46
 2  &"SPLIT",1
10  &"CHART",GGO(1)
```

where "SPLIT" is a use of the SPLITTER.TB command, and "CHART" is a direct use of AMPERCHART.TB.

REMEMBER, after adding Toolbox files, the Workbench must be removed before running the program!

The advantage of this command is the convenience of having one complete and self-contained program that doesn't have to access a second file on the disk. The disadvantages are:

1) The SPLITTER command takes up about 1K of free memory itself.

2) A program that has been split must be unsplit before editing. The SPLITTER gives you this capability, but the problem can become annoying, and;

3) The program will now take up much more room on the disk. For each program that uses this mode, there is another copy of AMPERCHART.TB residing on the disk, "hidden" within the program.

   This is the reason that the demo programs on the disk use the most simplified installation method possible, i.e. HELLO puts AMPERCHART.TB at the top of memory once (see Section 7.4) and then each demo calls AMPERCHART directly, without having each demo program contain AMPERCHART via the Workbench.

7.4 Stand-Alone Method: High Memory

By stand-alone usage, we mean using AMPERCHART without adding it to the program using the Workbench. You have probably already seen several examples of this option without realizing it; this is the method that is used to run most of the demo programs. Although not the preferred method, it an option when AMPERCHART is the only Applesoft command extension you are using in a program.

When using this option, AMPERCHART resides at the top of usable memory, and any Applesoft programs reside in their normal location beginning at $800. HIMEM is set just below the beginning of AMPERCHART in order to protect it from strings that may be created in the program.

AMPERCHART can be installed from within a program by including the following commands at the beginning of the program:

```
 1 HI=PEEK(115)+256*PEEK(116)-6131
 2 PRINT CHR$(4);"BLOAD AMPERCHART.TB,A";HI
 3 AH=INT(HI/256) : AL=HI-AH*256
 4 POKE 1014,AL : POKE 1015,AH
 5 HIMEM:HI
```

This magical incantation simply loads AMPERCHART as high as possible in memory, sets the ampersand vector so that it points to AMPERCHART, and then sets HIMEM to protect AMPERCHART.

The other versions of AMPERCHART (PLOT and PLOT+LABEL) can be installed by substituting the correct length of that particular version into Line 1, in place of the AMPERCHART.TB length (6131 bytes).

No matter which version is used, the appropriate HIMEM: and POKE's should be executed as one of the first operations in the running program, before any variables are defined.

The only advantage of this method is that when many programs are present on a diskette that is cramped for space, RELOCATE I/II.TB is not repeatedly stored on the diskette within each program.

The disadvantages arising from this option include the fact that if your program is larger than 6K, then SPLITTER or RELOCATE will be needed, and at that point, you would be better off using the RELOCATE option (assuming you're trying to conserve disk space).

With AMPERCHART tucked away in the high reaches of memory and the Applesoft program residing in its normal location starting at $800 there is only 6K to 14K available for an Applesoft program. Since the graphics pages reside in the region from $2000 to $6000 only programs that are smaller than 6K can use both pages. If a program is smaller than 14K then it can use just page 2. If these rules are not followed then the program will reside partly in the region of the Hi-Res pages, and any graphics commands that affect those pages may alter or destroy the program.

The other difficulty with the stand-alone method is that AMPERCHART makes sole use of the ampersand vector so other programs that normally use the ampersand vector (such as PLE or Apple's APA) will be disconnected. There only two ways around this: you can either use the ampersand vector for the utilities and use the CALL function to access your own commands, or you can use the Workbench to provide the links between all of your commands and use the AMPERSAND RESTORE command to restore the function of the utilities when your program ends.

In both cases, AMPERCHART is then loaded into the Apple for use within your programs. When installed in this way, AMPERCHART can only be removed or wrecked by executing either a FP, MAXFILES, or HIMEM: command, or by rebooting. The syntax for using AMPERCHART in this option is a little different in that the "CHART" (or other Toolbox command name) is not used. Since the ampersand points directly to AMPERCHART, graphic commands within the program would look like this:

```
 1 HI=PEEK(115)+256*PEEK(116)-6131
 2 PRINT CHR$(4);"BLOAD AMPERCHART.TB,A";HI
 3 AH=INT(HI/256) : AL=HI-AH*256
 4 POKE 1014,AL : POKE 1015,AH
 5 HIMEM:HI
10 &GGO(1)
20 &CLP((75,150,40,17)
30 &SCL(-100,100,-50,50)
```

Note that the AMPERCHART command directly follows the ampersand character.

7.5 Stand-Alone Method: Low Memory

This is sort of a hybrid approach. The idea is to BRUN RELOCATE I.TB or RELOCATE II.TB from either within a program, or even from the immediate mode. When done in this manner instead of as a Toolbox command, AMPERCHART.TB is loaded and your program is moved, but the Toolbox interface is not used, and the ampersand vector points right to AMPERCHART.

In this situation, the syntax corresponds to that used in the DEMO programs, and in the listing in the previous section.

Using this approach, a sample program might look like this:

```
 1 PRINT CHR$(4);"BRUN RELOCATE I.TB"
10 &GGO(1)
20 &CLP((75,150,40,17)
30 &SCL(-100,100,-50,50)
```

The main advantages of this approach are that AMPERCHART.TB and RELOCATE I.TB are not duplicated within many graphics programs on the disks, and that shorter graphics commands are used, i.e. the usual "CHART" prefix is not needed in each command line, which makes typing a little easier.

The disadvantage is that this precludes using the ampersand character for other Toolbox commands. If you only want to use AMPERCHART and none of the other Toolbox commands this is ok.

If however, you want to use other Toolbox commands with this installation method, you can use the CALL method for the other Toolbox commands as in this program:

```
  1 IR = PEEK(175) + 256*PEEK(176) - 203
  2 PRINT CHR$(4);"BRUN RELOCATE I.TB"
 10 &GGO(1)
 20 &CLP((75,150,40,17)
 30 &SCL(-100,100,-50,50)
100 CALL IR"TONE",P,D
```

7.10 Other Installation Methods

We have described four installation options that should suit nearly all applications. Other possibilities exist (most of which are not as useful) since AMPERCHART is absolutely relocatable; it will run anywhere you put it. For those of you interested, this is accomplished by having the program rewrite itself if it has moved from where it thinks it is. This leads to the slight delay that you may notice accompanying the first use of an AMPERCHART command after installation. Subsequent commands though will be executed with AMPERCHART's customary speed.

8. VERSIONS OF AMPERCHART

There are three versions of AMPERCHART that support standard Hi-Res graphics and two versions that support Double Hi-Res graphics included in this package. This allows you to choose just the subset of functions that you need for any particular application. The specifics of each of the standard versions are described below.

The primary Double Hi-Res version, DHR AMPERCHART is described in detail in Chapter 10, but also has the same instruction set as the standard AMPERCHART.TB. [Note that the offsets for locating the AMPERCHART internal locations are different for each version. See Chapter 11 for details.]

8.1 AMPERCHART.PLOT.TB - Basic Plotting

This is a small version of AMPERCHART that provides all of the functions necessary to perform basic plotting.

The labeling, axes and grid generation, filling, and arc drawing commands are NOT included in this version.

Length: 2746 bytes ($ABA)

Available Commands:

GGO	TMD	GMD	MIX	NMX	DSP	WRK
CLP	SCL	LMD	CLR	FCL	RVS	MOV
PLT	DRW	BIT	FRM	STU	LOC	

Deleted Commands:

AXS	GID	HLB	VLB	FIL	ARC	TSZ

To use this version, add the Toolbox file directly using the Workbench and use page 2 of Hi-Res graphics. RELOCATE I or II.TB and AMPERCHART JUMP FILE.TB should not be added when using this command set.

8.2 AMPERCHART.PLOT+LABEL.TB - Basic Plotting & Labeling

This version of AMPERCHART provides all of the functions available in the short version, but adds the labeling commands.

Length: 4184 bytes ($1058)

Available Commands:

GGO	TMD	GMD	MIX	NMX	DSP	WRK
CLP	SCL	LMD	CLR	FCL	RVS	MOV
PLT	DRW	BIT	HLB	VLB	FRM	
STU	LOC					

Deleted Commands:

AXS	GID	FIL	ARC	TSZ

To use this version, add the Toolbox file directly using the Workbench and use page 2 of Hi-Res graphics. RELOCATE I or II.TB and AMPERCHART JUMP FILE.TB should not be added when using this command set.

8.3 AMPERCHART.TB - Complete System

This is the standard version of AMPERCHART and provides all of the capabilities and commands listed in this manual. All of the demonstration programs make use of this version.

Length: 6141 bytes ($17FD)

Available Commands:

GGO	TMD	GMD	MIX	NMX	DSP	WRK
CLP	SCL	LMD	CLR	FCL	RVS	MOV
PLT	DRW	BIT	AXS	GID	HLB	VLB
FIL	ARC	TSZ	FRM	STU	LOC	

Deleted Commands: NONE

9. DOUBLE HI-RESOLUTION PLOTTING

The Apple //e and Apple //c computers have the capability of producing graphics with 560 dot horizontal resolution, twice the resolution obtainable from the Apple][or Apple][Plus. Some special commands are included on the AMPERCHART disk in order to support this mode. The software also supports several new features such as a reverse color (similar to XDRAW, but for HPLOT'ing) and automatic line type generation. This Chapter describes this new mode and the hardware and software required to use it.

9.1 Double Hi-Resolution Mode

The normal Apple Hi-Res display has a horizontal resolution of 280 pixels (pixels is short for 'picture elements'). The new Double Hi-Res mode has a horizontal resolution of 560 pixels, twice the old mode (hence the clever name: Double Hi-Res). The vertical resolution is the same for both modes: 192 pixels. This increased horizontal resolution allows you to put more information into your plots and graphs. For example, you now have a full 80 columns of space for standard size character labels.

The new mode can also be used to display up to 16 colors, albeit at a lower resolution of 140 pixels across. These colors are generated by producing certain patterns of pixels in each group of 4 pixels on the screen. This is similar to the standard Hi-Res mode where groups of two pixels (along with bit 7 in each byte) determine the color.

Because of conflicts with the 80 column card firmware, the software only supports a single Double Hi-Res page of graphics, Page 1. The address of the Double Hi-Res page is the same as for the standard Hi-Res Page 1 ($2000-$3FFF). The extra memory required to support this mode comes from the same memory locations in the extra 64K of RAM on the extended 80 column card. The requirements for keeping your program out of the Hi-Res area is therefore the same for Double Hi-Res work as it is for standard Hi-Res work.

9.2 Hardware Requirements

The following hardware is required to display Double Hi-Res on an Apple //e:

1. An Apple //e with a revision B or later motherboard. (The revision of your motherboard is given by the last letter in the serial number printed on the very back of the board, just behind slot C4.) If you have a revision A motherboard and can produce proof of purchase of any extended 80-column card, Apple Computer Inc. will provide you with a new motherboard.

2. An extended 80-column card (the version of the 80-column card containing an extra 64K of memory).

3. A jumper connecting the two molex pins on the extended 80-column board. This can either be the jumper that came with the card or an alligator clip placed securely across the molex pins. (Do not jumper this connection if you have a revision A motherboard or your computer will cease to function!)

Double Hi-Res can also be displayed on an Apple //c without any modification or additional hardware.

9.3 Double Hi-Resolution Software

There are four basic categories of software included on this disk that support the use of the Double Hi-Res mode:

1. The Double Hi-Res commands themselves;

2. Modified versions of AMPERCHART that can make use of the Double Hi-Res commands;

3. Double Hi-Res printer dump commands, and

4. Miscellaneous Double Hi-Res support software.

The commands were divided up this way, instead of being bundled into one or two larger command sets in order to provide you with the most flexibility possible. Despite this proliferation of commands, we think you will find the commands simple and easy to use. An overview of the commands is given below while the detailed description of their usage can be found on the following pages.

9.4 Double Hi-Res Command Sets

The Double Hi-Res commands are those commands which give you Double Hi-Res equivalents of the usual Applesoft Hi-Res commands such as HGR, HCOLOR, HPLOT, etc.

There are two primary implementations of these Double Hi-Res commands. The first is a pair of self-contained commands, DHR ROUTINES and DHR ROUTINES.TB.

The second is a special version, DHR MODIFIER.TB, that modifies the existing Applesoft in an Apple //e so that existing Applesoft programs will function in the Double Hi-Res mode with no particular changes.

DHR ROUTINES is specially designed to be used with DHR AMPERCHART, and is loaded automatically by the Toolbox file DHR RELOCATE.TB. NOTE: This is optional for the Apple //e, but is REQUIRED for Double Hi-Res operation on the //c.

DHR ROUTINES.TB is a stand-alone version of the Double Hi-Res commands for those cases where you don't need the DHR version of AMPERCHART and you do not want to wodify Applesoft (or cannot, as in the case of the Apple //c).

It is not compatible with the Double Hi-Res versions of AMPERCHART. This is because, as a "floating" Toolbox command, it's location is always changing, and thus inaccessable to DHR AMPERCHART.

DHR MODIFIER.TB modifies Applesoft in the Apple //e "on the fly" to provide for the new capabilities. This command can be used in conjunction with the first two Double Hi-Res versions of DHR AMPERCHART, i.e. DHR AMPERCHART.TB and DHR AMPERCHART. This cannot be used on the Apple //c.

To summarize then, the DHR (Double Hi-Res) commands exist only to provide the equivalent of commands like 'HGR' in the Double Hi-Res mode). They may be used by themselves, or combined with a DHR version of AMPERCHART if DHR chart functions are desired.

In general, for most applications, the most direct (and universal) method of implementing DHR graphics is to add DHR RELOCATE.TB and DHR JUMP FILE.TB to your program using the Workbench. This will install DHR AMPERCHART and DHR ROUTINES.

Section 9.8 in this chapter summarizes the various situations in which different versions of DHR ROUTINES and DHR AMPERHCHART may be, or are required to be used.

9.5 Double Hi-Res Commands

Any of the three main Double Hi-Res command sets support the same group of Double Hi-Res commands. These commands, which are nearly identical in form to the standard Hi-Res commands, are listed below:

HGR - Same effect and idiosynchrasies as HGR, but in Double Hi-Res mode. This also turns on the 80-column mode of the Apple //e so that mixed graphics and text look OK. Note that the clear screen algorithm is slow; it was written for memory efficient operation, not speed.

HPLOT, DRAW, XDRAW - All the same as before, but in Double Hi-Res mode. Note that now the legal range of x-coordinates is from 0 to 559.

HCOLOR= - Similar to before, but now the colors go from 0 to 16. On this scale 0 is black, 15 is white, and the intermediate values are the colors normally available on the Lo-Res screen. Specifying HCOLOR=16 selects the new 'color', REVERSE. REVERSE plots in an opposing color to whatever the background is, similar to XDRAW.

Note that similar to standard Hi-Res, a single vertical line drawn in a color may or may not appear on the screen depending on the line's location relative to the groups of four pixels used to generate color. To insure that you get a vertical colored line you have to draw 4 vertical lines on the screen, each adjacent to the next, and all aligned with the screen nibbles. The default color is white.

LNTYP=linetype - The only truly new command. This command allows you to choose the linetype that the Apple will use when drawing. The linetype variable can either be in the range 0-7, with results as shown in the table, or it can be a negative number between the ranges of -16777215 (=2^24-1) and -1.

If a negative value is specified for linetype, then the linetype is generated from the binary representation of the absolute value of the specified linetype (e.g. LNTYP=-1 would specify a single dot followed by 23 spaces as the linetype).

Due to the way Applesoft calculates formulae, if you want to specify a linetype of -2^23, you must force the exponent to be calculated first: LNTYP = - (2 ^ 23). The linetype stays in effect until a new linetype is specified. The default linetype is solid.

linetype #	linetype	
0	----------	solid
1	----------.....	long dashed
2	------.....------.....	dashed
3	---...---...---...---...	short dashed
4	-..-..-..-..-..-..-..-..	dotted
5	----------....----....	weird
6	---------...---...---...	weirder
7	-----....----...---..--.	weirdest
<0	User defined by ABS(#)	

Note: Combining non-zero linetypes and color will generate some very interesting, if not useful, AMPERCHART fills (see the demo, SATURN).

9.6 DHR ROUTINE Command Options

If DHR MODIFIER.TB (or DHR RELOCATE.MOD.TB) is used, Applesoft is directly modified and all Applesoft graphics commands work in their "native" state. A sample program using DHR MODIFIER.TB via the DHR RELOCATE.MOD.TB command might look like this:

```
 1 CALL PEEK(175) + 256 * PEEK(176) - 46
 2 &"RELOCATE.MOD"
10 &"CHART",GCO      [NOT GCO(1)]
20 HCOLOR=15
30 HPLOT 0,0 TO 100,100
```

If DHR ROUTINES or DHR ROUTINES.TB are used, then the DHR commands must be accessed by adding them as a Toolbox file using the Workbench.

If DHR ROUTINES are used, they are usually loaded, along with DHR AMPERCHART, by the command DHR RELOCATE.TB. In that case, all calls to BOTH AMPERCHART and the DHR ROUTINES are made through the DHR JUMP FILE.TB that is added to your program. In this case, a program would look like this:

```
 1 CALL PEEK(175) + 256 * PEEK(176) - 46
 2 &"RELOCATE"
10 &"CHART",GGO
20 &"CHART",HCOLOR=15
30 &"CHART",HPLOT 0,0 TO 100,100
```

Note that the AMPERCHART command name "CHART" is used to call both AMPERCHART and the DHR commands.

If DHR ROUTINES.TB are used AMPERCHART cannot be used. The only reason for using this command set would be if you had an Apple //c, and did not want to use AMPERCHART functions in a program.

For detailed information on the various Double Hi-Res modules, see the command descriptions later in this chapter.

For more information on Double Hi-Res, see the articles in the July 1983 and September 1983 issues of Softalk, although the best way to learn about Double Hi-Res is just to try it! In particular, try experimenting with vertical and nearly vertical lines.

9.7 Double Hi-Res Versions of AMPERCHART

There are also three Double Hi-Res versions of AMPERCHART included on this disk: DHR AMPERCHART, DHR AMPERCHART.MOD and DHR AMPERCHART.MOD.TB.

DHR AMPERCHART is a full implementation of AMPERCHART as described in Chapters 1,4 and 8. It is a fixed location version designed to be loaded and run at location $800. This fixed version is shorter than 6K bytes in length and hence fits under Hi-Res page 1. It expects to find DHR ROUTINES loaded at $4000, and is usually used with DHR RELOCATE and DHR JUMP FILE.TB.

DHR AMPERCHART.MOD and DHR AMPERCHART.MOD.TB are versions of AMPERCHART that require Applesoft to be modified using DHR MODIFIER.TB on an Apple //e.

DHR AMPERCHART.MOD is a fixed version to be loaded at $800 and is used with DHR RELOCATE.MOD.TB.

DHR AMPERCHART.MOD.TB is a position independent command and hence is fully compatible with the Toolbox system. Because your program will be larger than 6K though, some sort of memory management is required such as adding SPLITTER.TB.

If you have an Apple //c, or want to create the most universally compatible DHR software, add DHR RELOCATE.TB and DHR JUMP FILE.TB with the Workbench. When your program runs, RELOCATE will load the file DHR AMPERCHART and DHR ROUTINES from the disk.

If you have an Apple //e, and want to use a modified version of Applesoft (DOS 3.3 only), then use the Workbench to add DHR RELOCATE.MOD.TB to your program. When run, RELOCATE will modify Applesoft using DHR MODIFIER.TB and load DHR AMPERCHART.MOD at $800 after moving your program.

9.8 Summary of Installation Options

If you want to avoid a lot of decisions, the best approach is to just add DHR RELOCATE and DHR JUMP FILE.TB to your program using the Workbench, and let these modules take care of everything. This setup will work on all Apples and the only requirement is that you make sure that DHR AMPERCHART and DHR ROUTINES are present on the disk when the program is run since these files are loaded by DHR RELOCATE.TB.

Other options have been discussed though, which have different advantages in terms of memory usage. A summary is presented here. The main decisions to be made are:

1. What system will the software run on? (Apple //e with or without 16K RAM available, or Apple //c?)

and

2. Will DHR AMPERCHART also be used?

A. If an Apple //c and DHR AMPERCHART, then the only option is to use DHR RELOCATE.TB as described above.

B. If only DHR ROUTINES are desired on a //c (no AMPERCHART), then DHR ROUTINES.TB can be added using the Workbench.

If an Apple //e, then the next question is whether the upper 16K of RAM is available.

D. Ordinarily this area would be available, and you should add DHR RELOCATE.MOD.TB and DHR AMPERCHART JUMP FILE.TB to your program using the Workbench, and make sure that DHR MODIFIER.TB and DHR AMPERCHART.MOD are present on the disk.

E. If the 16K is not available, for instance because you are using a moved DOS, then you should use the DHR RELOCATE.TB and DHR AMPERCHART JUMP FILE.TB as described in Chapter 3 of this manual.

9.9 Double Hi-Res Dump Commands

A set of printer dump commands are provided to support the Double Hi-Res mode only on the Epson and Apple ImageWriter printers. These commands work basically the same as the standard printer commands, but will dump what you see on the Double Hi-Res display on your printer. These commands will not work on any other type of printer.

9.10 Miscellaneous Support Commands

Included are several useful Double Hi-Res utilities that allow you to turn the Double Hi-Res display on, move the Double Hi-Res screens around, and use the additional 64K of memory in the Apple //e or //c for Hi-Res image storage and save and restore graphic windows. An additional command is included allowing your program to identify the type of Apple that it is running on so the program can determine if Double Hi-Resolution displays are possible.

9.11 Double Hi-Res Demonstration Programs

There are several programs contained on the AMPER-CHART diskette that will demonstrate the capabilities of the Double Hi-Res system. To run these programs, just boot on side 2 of the Chart 'n Graph diskette, and then select DEMONSTRATION PROGRAMS in the main menu.

This will set up the Double Hi-Res mode, install a fixed version of DHR AMPERCHART and move your program to protect Hi-Res page 1. This will work on an Apple //e as well as an Apple //c.

Some of these are transcriptions of standard AMPERCHART demos (e.g. LABEL TEST, BAR CHART, and CHAPMAN PLOT), and some are newly designed to demonstrate capabilities of the Double Hi-Res (e.g. APPLE STOCK DEMO, COLOR DEMO).

The other demonstration program is IDENTIFICATION DEMO which will tell you the configuration of the Apple that you are running on (as if you didn't know).

>> DHR RELOCATE.TB <<

and

>> DHR AMPERCHART JUMP FILE.TB <<

FUNCTION: These two commands are used together on an Apple //e or Apple //c to implement Double Hi-Res AMPERCHART and Double Hi-Res BASIC commands.

LENGTH: DHR RELOCATE.TB: 338 bytes ($152)
DHR JUMP FILE.TB: 3 bytes ($3)

SYNTAX: &"NAME1": &"NAME2",command [,variables]

SAMPLE: &"RELOCATE":&"CHART",GCO:&"CHART",HCOLOR=15

HOW TO USE IT: These commands are added using the Workbench when both AMPERCHART and/or Double Hi-Res commands are desired. This setup works on both the Apple //e and //c computers.

DHR AMPERCHART JUMP FILE.TB is also used by itself as a the Toolbox compatible jump to DHR AMPERCHART.MOD when this command set is installed using DHR RELOCATE.MOD.TB.

LIMITATIONS: The commands DHR AMPERCHART and DHR ROUTINES must be present on the active disk when the DHR RELOCATE.TB command is called within a running program.

>> DHR AMPERCHART <<

by Andy Scheck

FUNCTION: A full implementation of AMPERCHART, designed to be loaded and run using DHR RELOCATE.TB and DHR JUMP FILE.TB. This command set is compatible with both the //e and //c.

LENGTH: 5858 bytes ($16E2)

SYNTAX: &"NAME",DHR command [,variable list]

SAMPLE: &"DHR AMPERCHART",GCO
&"DHR AMPERCHART",PLT(UX,UY)

=> See Chapters 3 and 4 for a complete description of these and all of the rest of the AMPERCHART commands.

HOW TO USE IT: This version of AMPERCHART performs the same functions as the previously described version. The only difference is that this version must be used with DHR ROUTINES installed at $4000. For this reason it is completely compatible with all Apple // family computers. It is only compatible with the Workbench by using the jump file included on the back side of the disk, DHR JUMP FILE.TB.

To install this command in the usual manner, append DHR RELOCATE.TB and DHR JUMP FILE.TB to your program using the Workbench.

This setup command protects any existing Applesoft program before installing both DHR AMPERCHART and DHR ROUTINES into memory and setting up the ampersand vector. This command is called by the HELLO program on the back of the disk to set up the machine for the DHR demonstration programs.

LIMITATIONS: See DHR ROUTINES.

>> DHR ROUTINES <<

by Rick Chapman

FUNCTION: Implements all of the new Double Hi-Res commands in a standalone, fixed-location command set that is compatible with both the Apple //c and //e. This command set must be used with DHR AMPERCHART.

LENGTH: 1162 bytes ($48A)

SYNTAX: &"NAME",command (,variables)

SAMPLE:
```
&"DHR",HGR
&"DHR",HPLOT X,Y TO X1,Y1
&"DHR",DRAW SH AT X,Y
&"DHR",XDRAW SH AT X,Y
&"DHR",HCOLOR=15
&"DHR",LNTYP=3
```

=> Execute the same Double Hi-Res commands as described in the last section, but in a standalone package that's compatible with the Apple //c.

HOW TO USE IT: This command set must be used in conjunction DHR AMPERCHART. It cannot be used by itself. To install the command set in the usual manner, just add DHR RELOCATE.TB and DHR JUMP FILE.TB to your program using the Workbench.

This setup command will relocate the current Applesoft program to $4500, install DHR ROUTINES at $4000 and DHR AMPERCHART at $800. This is the command called by the HELLO program on the back side of the Chart 'n Graph Toolbox diskette.

The calling syntax shown above is valid if a Toolbox jump file is used to access the command. Such a command is included on the disk as DHR JUMP FILE.TB.

If the command is called directly (as set up by BRUNning the DHR RELOCATE.TB command) then the syntax is the same but without the command name and first comma (e.g. &LNTYP=4 or &HPLOT 10,10 to 100,100). This is the calling syntax used in the DHR demo programs on the disk. Otherwise, the instructions and their actions are identical to those described in Section 9.3.

This command does not make use of the upper 16K of memory in the Apple //e or //c and thus is fully compatible with programs that do use that area, such as DOS movers.

LIMITATIONS: The limitations of this command are as follows:

1. All of the standard Hi-Res error messages are supported.

2. A single zero page location is used by these commands, location $E3. The use of this location should be avoided while the Double Hi-Res commands are being used. BASIC-only programmers needn't concern themselves with this.

3. When converting programs written for use with the MODIFIED commands, all of the LNTYP= commands must be retyped completely. See the text on LNTYP in the DHR MODIFIER.TB section for details.

4. The command must be used with DHR AMPERCHART which occupies the 6K byte region below Double Hi-Res page 1. Thus, when this command is installed, any Applesoft program must be moved to begin at location $4500 (this is done automatically by the setup command.) This reduces the space available for the Applesoft program by over 15K bytes.

>> DHR DISPLAY.TB <<

by Rick Chapman

FUNCTION: To switch on the display of Double Hi-Res graphics, without remembering or using a dozen different pokes and peeks.

LENGTH: 43 bytes ($2B)

SYNTAX: &"NAME"

SAMPLE: &"DISPLAY"

=> Displays Double Hi-Res page 1 and turns on the 80 column mode for text display.

HOW TO USE IT: This command simply switches on the Double Hi-Res display, without going through a seemingly endless incantation of peeks and pokes.

As a side effect, the 80 column mode is turned on and the 80 column support software is activated. If the text display were to be left in 40 column mode, then the text in mixed graphics and text mode would be printed in only every other space. Thus if you want Double Hi-Res graphics and 40 column text, you can use this command followed by a PRINT CHR$(27);CHR$(15) command.

LIMITATIONS: The command only works on an Apple //e with an extended 80 column card installed or an Apple //c.

>> DHR LOAD.TB <<

by Rick Chapman

FUNCTION: To load a double hi-res image file saved by DHR SAVE.TB into Page 1 of memory.

LENGTH: 492 bytes ($1EC)

SYNTAX: &"NAME","filename",1

SAMPLE: &"DHR LOAD","PIC1",1

=> Loads the double hi-res image S-file named PIC1 into Page 1.

HOW TO USE IT: This command is used to load double hi-res images from the disk. The double hi-res image files must be standard format, 65-sector S-files.

The syntax, as shown above, is the same as for the standard hi-res load module, BLOAD PIC.TB. The filename can be any legal DOS file name. If the named file does not exist, a FILE NOT FOUND error will be printed. The command only works with Page 1 of double hi-res and so the pagenumber variable following the file name must be set to 1. Any other value will cause an error.

The command has to shift the double hi-res image around in memory as it is being loaded in, so the double hi-res display may look a little strange during the load process. In the end though, everything sorts itself out and the stored image is accurately restored to the screen.

All standard DOS error messages are supported. The command will work with either standard Apple DOS or the special DOS that comes on the Chart 'n Graph Toolbox diskette.

LIMITATIONS: The file is always loaded from the last accessed disk drive. There is no way to specify the desired drive number. This command may not work with some modified versions of DOS.

>> DHR SAVE.TB <<

by Rick Chapman

FUNCTION: To save a double hi-res image in a standard format DOS file.

LENGTH: 576 bytes ($240)

SYNTAX: &"NAME","filename",1

SAMPLE: &"DHR SAVE","PIC1",1

=> Saves the double hi-res image on Page 1 as a standard format DOS file named PIC1.

HOW TO USE IT: This command is used to save double Hi-Res images out to disk. The images are stored in a special S-type file which are compatible with other double hi-res graphics programs.

The syntax, as shown above, is the same as for the standard hi-res save module, BSAVE PIC.TB. The filename can be any legal DOS file name. If the named file does not exist, one will be automatically created. The command only works with Page 1 of double hi-res and so the pagenumber variable following the file name must be set to 1. Any other value will cause an error.

The file created by this command will be a 65 sector S-type file. This file cannot be BLOADed. The only way to load this file back into memory is to use the DHR LOAD.TB command or some other double hi-res graphics package.

This command shuffles portions of the double hi-res image around in memory as it is saving the image to disk. Advanced users should be aware that image information is temporarily stored in Auxillary memory in the location normally used for Page 2 graphics ($4000-$5FFF). Thus this area must be reserved if you are using the Auxillary memory as a RAM Disk.

All standard DOS error messages are supported. The command will work with either standard Apple DOS or the special DOS that comes on the AMPERCHART diskette.

LIMITATIONS: The file is always saved to the last accessed disk drive. There is no way to specify the desired drive number. This command may not work with some modified versions of DOS.

>> DHR WINDOW.TB <<

by Rick Chapman

FUNCTION: This command can store a rectangular area of the Double Hi-Res screen into an integer array and then later restore that area. This provides the basis for writing programs that use windowing or pull down menus or it can be used for simply moving image areas around on the screen.

LENGTH: 461 bytes ($1CD)

SYNTAX: &"WINDOW",dir,xbgn,xend,ymin,ymax,1,
intarray
&"WINDOW",aexpr,aexpr,aexpr,aexpr,aexpr,1,
intarray

SAMPLE: &"WINDOW",1,0,3,151,180,1,X%(0)

=> Moves the contents of the specified rectangular area of Double Hi-Res page 1 into the integer array X% and then blacks out that area. The specified area is the first 4 bytes (0 through 3) of lines 151 through 180, that is $0 \leq X \leq 27$ and $151 \leq Y \leq 181$.

&"WINDOW",0,6,9,141,170,1,X%(0)

=> Moves the contents of the integer array X% back into an area of Double Hi-Res page 1 that is 42 pixels to the left and 10 pixels below the area that was stored in the instruction above.

HOW TO USE IT: This command will either move a rectangular area into an integer array or move the contents of that array back onto an area of the screen. The command is quite fast and was designed to make it easy to develop sophisticated looking graphic displays and menus.

All of the variables must be specified, none are optional. The direction variable determines the direction of the data flow according to the following table:

direction	action
0	array -> graph
1	graph -> array with clear
2	graph -> array without clear

A direction variable of 1 moves the specified area into the array and then clears the specified area on the screen to black. This allows the user to fill in the blackened area with anything you might want, such as a menu of options, and then later restore the area to its original contents. A direction variable of 2 performs the same function but does not clear the area. This can be used if you want to make a copy of the area onto another part of the screen. A direction of 0 restores the contents of the area on the screen without disturbing the contents of the array.

The xbgn and xend variables literally specify the beginning and end bytes in a each line of the area to be affected. Each byte contains 7 pixels so this command will only move areas in increments of 7 pixels wide. This was done to minimize memory usage and maximize speed, but turns out to be quite handy since the characters used in AMPERCHART are exactly 7 pixels wide (including the space around them). The xbgn and xend variables can range from 0 to 79 and xend must be greater than xbgn.

The ymin and ymax variables specify the lower and upper boundaries (in standard AMPERCHART coordinates) of the area to be affected, respectively. These variables must lie in the range 0 to 191 and ymin must be less than ymax.

The page variable (see WINDOW.TB) must be set to 1. Any other value will result in an error. This variable was included to make the syntax compatible with the syntax of the WINDOW.TB command.

The array that is used by this command must be specified as an integer array having a single dimension. The subscript to the array must be included and should be set to zero. If an integer array of that name has already been dimensioned, then the command makes sure the array is large enough to hold all of the data being moved and then peforms the specified action. If the array is found to be too small, even if the direction of the transfer is from array to graph, an error message is printed and the command will terminate. If no such integer array is found, one is created automatically. Thus you do not have to pre-dimension the arrays used by this command; it is in fact safer and simpler not to and let the command do all the work.

LIMITATIONS: The command only moves areas beginning and ending at byte boundaries. This command does not work with the Double Hi-Res commands included elsewhere on this disk. The following error messages are supported:

SYNTAX ERROR - The syntax of the command is incorrect.

ILLEGAL QUANTITY - One of the specified variables is out of range.

TYPE MISMATCH - The specified array has already been declared and is either not an integer array or has to many dimensions.

OUT OF MEMORY - There is not enough room in memory to create the specified array.

TECHNICAL NOTES: The command saves the data from the screen into the integer array by rows, proceeding from the left bottom corner of the area to the top right corner. If you want to use this command to get a portion of the screen into an array to manipulate, remember that the pixels on the screen are stored in reverse order in each byte, thus the least significant bit of a screen byte is displayed to the left of bit 6 of that same byte. Also remember that bit 7 of each byte is not displayed, it is used only to determine color.

The integer arrays that are created by this command can be very large, up to 8K bytes. To figure the size of the array created, use the approximate formula: # of bytes in the array = (1+xbgn-xend)*(1+ymax-ymin)*2+7.

>> DHR TRANSFER.TB <<

by Rick Chapman

FUNCTION: This command allows you to transfer Hi-Res images from main to auxillary memory and back again. This provides both the capability of storing and retrieving up to 5 Hi-Res images in the auxillary memory of the Apple //e (or //c) and also an alternative means of using DOS to save or load Double Hi-Res images.

LENGTH: 79 bytes ($4F)

SYNTAX: &"NAME",source,destination,direction
&"NAME",avar,avar,avar

SAMPLE: &"TRANSFER",2,4,1

=> Moves the Hi-Res page 2 in main memory to an area of auxillary memory labelled page 4.

HOW TO USE IT: This command allows you to store and retrieve Hi-Res images using the extended 80 column card's memory. Up to five Hi-Res pages can be stored in the auxillary memory (and 3 more can be stored in main memory) and recalled faster than a speeding bullet.

The first variable designates the source page for the transfer. It must be in the range 1 to 5. A value of 1 indicates that the source range in memory is $2000 to $4000, the exact region of Hi-Res page 1. Likewise page 5 resides from $8000 to $A000, a region in main memory that will probably include a portion of DOS.

The second variable designates the destination page, in the exact same manner as the source variable. The destination variable must also be in the range from 1 to 5.

The third variable determines the direction of the transfer according to the folowing:

direction = 0 : transfers CARD ==> MAIN
direction = 1 : transfers MAIN ==> CARD.

Be aware that careless use of this command can clobber your program or DOS or both. The flexibility of using main memory for storing images (by first transferring to the card, then back into a different location) was included to make the command as general as

possible. Unfortunately it allows you to overwrite whatever is in memory, if you make a mistake in the direction or page number. Moral: BE CAREFUL!

This command also allows you to save and restore Double Hi-Res images onto disk. If Hi-Res page 2 is available in main memory then the procedure is as follows:

```
100 &"TRANSFER",1,2,0     :REM TRANSFER PAGE 1 FROM CARD
                                TO PAGE 2 OF MAIN
110 PRINT CHR$(4);"BSAVE PIC,A$2000,L$3FFB"
.
.
300 PRINT CHR$(4);"BLOAD PIC,A$2000"
310 &"TRANSFER",2,1,1     :REM TRANSFER PAGE 2 FROM MAIN
                                TO PAGE 1 OF CARD
```

Line 100 performs the transfer to page 2 of main memory, while line 110 saves both Hi-Res pages in main memory. The use of a length of $3FFB instead of $4000 makes no difference in the stored picture, but reduces the number of sectors that the picture takes up on the disk by one. To retrieve the Double Hi-Res image, simply reverse the procedure as shown in lines 300 & 310.

If you are constantly experiencing a memory crunch as I usually am, you can use the following modified technique (resulting in two disk files instead of one) to save and then restore a Double Hi-Res image.

```
100 PRINT CHR$(4);"BSAVE PIC1,A$2000,L$1FFB"
110 &"TRANSFER",1,1,0     :REM TRANSFER PAGE 1 FROM CARD
                                TO PAGE 1 OF MAIN
120 PRINT CHR$(4);"BSAVE PIC2,A$2000,L$1FFB"
.
.
300 PRINT CHR$(4);"BLOAD PIC2,A$2000"
310 &"TRANSFER",1,1,1     :REM TRANSFER PAGE 2 FROM MAIN
                                TO PAGE 1 OF CARD
320 PRINT CHR$(4);"BLOAD PIC1,A$2000"
```

LIMITATIONS: The command only works on an Apple //e with an extended 80 column card installed or an Apple //c.

>> DHR ROUTINES.TB <<

by Rick Chapman

FUNCTION: Implements all of the new Double Hi-Res commands in a stand-alone or Toolbox compatible package, without modifying Applesoft.

LENGTH: 1361 bytes ($551)

SYNTAX: &"NAME",command (,variables)

SAMPLE:
```
&"DHR STANDALONE",HGR
&"DHR STANDALONE",HPLOT X,Y TO X1,Y1
&"DHR STANDALONE",DRAW SH AT X,Y
&"DHR STANDALONE",XDRAW SH AT X,Y
&"DHR STANDALONE",HCOLOR=15
&"DHR STANDALONE",LNTYP=3
```

=> Execute the same Double Hi-Res commands as described in the last section, but in a completely stand-alone, Toolbox-compatible package.

HOW TO USE IT: This command can be used in conjunction with the Workbench or by itself. To use the command in conjunction with the Workbench, follow the standard Toolbox command installation procedure. The calling syntax for this case is exactly as shown above.

To use the command without the Workbench follow the directions in Section 5.1. The syntax for this case lacks the command name, only the command name and variables are included. For example, the HPLOT command syntax is: &HPLOT X1,Y1 TO X2,Y2.

This command does not make use of the upper 16K of memory in the Apple //e or //c and thus is fully compatible with programs that do use that area.

The DHR versions of AMPERCHART are not designed to work with this command. If you want to use AMPERCHART in the Double Hi-Res mode, you must use either DHR MODIFIER.TB or DHR ROUTINES.

LIMITATIONS: The limitations of this command are as follows:

1. All of the standard Hi-Res error messages are supported.

2. A single zero page location is used by these commands, location $E3. The use of this location should be avoided while the Double Hi-Res commands are being used. BASIC-only programmers needn't concern themselves with this.

3. When converting programs written for use with the MODIFIED commands, all of the LNTYP= commands must be retyped completely. See the text in the section on DHR MODIFIER.TB for details.

4. This command is not compatible with the Double Hi-Res versions of AMPERCHART.

>> DHR MX-80.TB <<
>> DHR MX-100.TB <<
>> DHR FX-80.TB <<
>> DHR IMAGEWRITER.TB <<

by Andy Scheck and Rick Chapman

FUNCTION: These are a set of graphics dump commands for Epson and Apple printers that are capable of printing the contents of Double Hi-Res page 1.

LENGTH:

DHR MX-80.TB	302 bytes	($12E)
DHR MX-100.TB	302 bytes	($12E)
DHR FX-80.TB	323 bytes	($143)
DHR IMAGEWRITER.TB	418 bytes	($1A2)

SYNTAX: &"NAME",1,t [,mode]
&"NAME",1,aexpr [,aexpr]

SAMPLE: &"DHR MX-80",1,10

=> Prints the contents of Double Hi-Res page 1 to the printer with a left margin of one inch (10 tenths of an inch).

&"DHR FX-80",1,5,4

=> Prints the contents of Double Hi-Res page 1 to the printer with a 1/2" left margin using 8-pin bit image mode 8 (CRT Graphics with 640 dots horizontally per 8".

HOW TO USE: The above commands are similar in function to the standard AMPERCHART printer commands, but print the Double Hi-Res page 1 instead of standard Hi-Res. In order to maintain compatibility of format with the standard AMPERCHART dump commands, the first variable is the page number and MUST be a 1. Attempts to specify page 2 will result in an error.

The MX series commands come only in a single size that prints a full 560 dots across, even on an MX-80. The tab variable is limited to a range of 0-33 for the MX-80 command and 0-90 for the MX-100 command. That in fact is the only difference between the two commands.

The FX dump command works on either an FX-80 or an FX-100 and also allows you to select a tab variable. The tab variable can range fom 0-255 on the FX command since the size of the printout is variable. If you specify a tab variable that cuts off part of the picture, then part of the picture is not printed (the printer behaves quite nicely).

If you select an extremely large tab variable (say trying to tab out 10 inches on 8-1/2 inch paper) the printer will quite correctly ignore your command and left justify the image.

The FX command also supports an optional variable that specifies the 8-pin bit image mode to be used by the printer. This determines the horizontal density of the plotted dots and hence the width of the plot. The acceptable range of values for this mode is from 0-6. See the Epson manual for more information on these modes. If this variable is not specified the command defaults to mode 1 (standard-speed, dual-density) resulting in a plot width of 4-2/3 inches.

Users of the new FX printers should note that the MX-100 commands will work on both the FX-80 and FX-100. Also, users of the FX-100 can use the DHR FX-80.TB.

The Imagewriter command prints a single size page. The tab variable is limiited to the range of 0-41. The command only works with a Super Serial Card Interface.

These commands do not require that the Double Hi-Res commands be installed, and in fact switch the upper memory back to ROM while they are active. Naturally, they restore everything to its previous condition before finishing, but if you hit CTRL-RESET to exit the command, then the ROM will be left switched in. If you were using the Double Hi-Res commands at the time simply type POKE -16253,0 to reset the RAM card.

The commands support the Double Hi-Res mode and will only work on a properly configured Apple //e. They are placed on the front of the disk, instead of the back in order to allow the PRINTER CONFIGURE program to have direct access to them in case you need to reconfigure them. Note that the PRINTER CONFIGURE program has been modified to work with these commands; the old version cannot be used to update these commands.

LIMITATIONS: These dump commands come setup for a standard Epson APL parallel interface card (or Apple or Tymac or etc.). If you have a different interface card you must install the correct interface using the PRINTER CONFIGURE program. To use this program follow the directions in section 5.11 of this manual.

Only Double Hi-Res page 1 can be printed.

>> DHR RELOCATE.MOD.TB <<

and

>> DHR JUMP FILE.TB <<

FUNCTION: These two commands are used together only on an Apple //e under DOS 3.3 to implement Double Hi-Res AMPERCHART and Double Hi-Res BASIC commands.

LENGTH: DHR RELOCATE.MOD.TB: 360 bytes ($168)
DHR JUMP FILE.TB: 3 bytes ($3)

SYNTAX: &"NAME1": &"NAME2",command [,variables]

SAMPLE: &"RELOCATE": &"CHART",GCO
HCOLOR=15

HOW TO USE IT: These commands are added using the Workbench when both AMPERCHART and Double Hi-Res commands are desired. This setup works only on Apple //e computers.

LIMITATIONS: The commands DHR AMPERCHART.MOD and DHR MODIFIER.TB must be present on the active disk when the DHR RELOCATE.MOD.TB command is used within a running program.

>> DHR MODIFIER.TB <<

by Rick Chapman

FUNCTION: Modifies Applesoft to support the full use of the Double Hi-Res graphics mode either directly from an Applesoft BASIC program or from a modified version of AMPERCHART. This command only works on the Apple //e, it is INCOMPATIBLE with the Apple //c.

LENGTH: 966 bytes ($3C6)

SYNTAX: &"NAME"

SAMPLE: &"DHR MODIFY"

=> Modifies Applesoft and the Monitor to provide full graphics support for the Double Hi-Res mode.

HOW TO USE IT: This command is designed to modify Applesoft and the Monitor to provide support for the Double Hi-Res mode. The command rewrites sections of Applesoft and the Monitor on the fly, swapping sections of code that are contained within itself (originally the Double Hi-Res commands) with the appropriate sections of Applesoft code. If it is called a second time, it will swap the sections back so that the original version of Applesoft is restored and the Double Hi-Res commands are again imbedded within the modifier command.

Because of differences in the internal structure of the Apple //e and //c, this command will not function on an Apple //c. Double Hi-Res support for the //c is provided by using DHR ROUTINES.

Once Applesoft and the Monitor have been modified by this command, all of the Double Hi-Res commands listed in Section 9.5 are available. They replace the standard Applesoft Hi-Res and can be used just as you would use those standard commands.

There are basically two ways of using this command; either you can use it once, installing the Double Hi-Res commands and then leaving them, or you can use the command to swap the Double Hi-Res commands in and out of Applesoft as needed.

Single Time Installation

If you are using an unmodified version of Applesoft and the Monitor (no fast garbage collectors, error fixes or other patches installed), AND you are not going to use any low resolution graphics commands (GR, COLOR=, PLOT, HLIN, VLIN, and SCRN) or tape commands LOAD (tape), SAVE (tape), and SHLOAD) then you probably want to install the Double Hi-Res commands only once.

The command can be used to install the Double Hi-Res commands by either BRUNing it or BLOADing it and then CALLing it. For example, from the keyboard, you can simply type BRUN DHR MODIFIER.TB. This will load the command at $2000 and then execute it. From within a program that is going to use Hi-Res graphics it can be run by including the statements:

```
1120 PRINT CHR$(4);"BLOAD DHR MODIFIER.TB"
1130 CALL (8192)
```

where the statements can be placed anywhere in the program. Note that this example assumes that portions of your program do not lie in Hi-Res page 1, a pre-requisite for the use of the Double Hi-Res page 1 (see the discussion in Chapter 4).

Once installed, the commands will remain in memory until the machine is turned off, you hit the CONTROL-OPEN APPLE-RESET sequence, or you or your program modifies the 16K RAM card area (the top 16K of RAM in the Apple //e). If the RESET button is pressed, or any cold or warm starts are performed, the commands are still in memory, but no longer active. To reactivate the commands simply type (or have a line in your program that performs the function):

```
POKE -16253,0 : REM REACTIVATES RAM FOR READING
```

In fact, it is probably a good idea to include such a line at the beginning of each program that you are going to use while the Double Hi-Res commands are installed.

Code Swapping Installation

If you are using a modified version of Applesoft or the Monitor and/or you want to use some of the Lo-Res graphics or tape commands then you must use the command to swap the Double Hi-Res commands in and out of Applesoft.

To do this, simply install the command using Routine Machine or at any convenient fixed location and call the command each time you want to swap the features you are using in Applesoft. For example, if you wanted to mix Double Hi-Res and Lo-Res commands in the same program (heaven knows why, but it's just an example) then you might end up with something like what's shown below:

```
1 CALL PEEK(175) + 256 * PEEK(176) - 46
2 GR                        : REM ERASE LO RES SCREEN
3 &"MODIFY"                 : REM INSTALL DOUBLE HI RES
                                  ROUTINES
4 HGR                       : REM SET DOUBLE HI RES & ERASE
                                  SCREEN
5 HPLOT 10,10 TO 500,90 : REM DRAW LONG LINE
6 &"MODIFY"                 : REM RESTORE LO RES ROUTINES
7 PLOT 10,10                : REM PLOT POINT IN LO RES
```

Despite the fact that this is a ridiculously useless program, it illustrates the main idea: when you first want to use the Double Hi-Res commands just CALL the command, and then when you want to make use of some capability in Applesoft that may have been wiped out by the Double Hi-Res commands just CALL the command a second time. By now you should realize that a third CALL would restore the Double Hi-Res commands and a fourth CALL would restore Applesoft, etc.

Implementation Details

The above instructions are implemented within the confines of Applesoft by using portions of Applesoft that were previously dedicated to the Lo-Res plotting commands, the tape commands and the monitor disassembler. For this reason, any call to the Lo-Res or tape commands while the Double Hi-Res commands are installed in Applesoft will cause an ILLEGAL INSTRUCTION ERROR to be generated. No error message will occur if you attempt to use the disassembler while these commands are installed, but the resultant output will be unintelligible.

The installation of these commands into Applesoft requires the use of the 16K RAM card area of memory in the Apple //e. Hence, they are not compatible with other programs (except other modified Applesofts) that use that area. This includes RAM card versions of DOS and ProDOS from Apple Computer Inc.

LIMITATIONS: The limitations to these commands are as follows:

1. These commands are absolutely not compatible with the Apple //c. They will only function on a suitably equipped Apple //e.

2. The commands replace the Lo-Res, tape and disassembler commands while they are installed, though they can be swapped in and out of the system to avoid conflicts. These replaced commands should not be called while the Double Hi-Res commands are installed.

3. The commands make use of the upper 16K of memory in the Apple //e and so are not compatible with DOS movers, ProDOS, or other programs which use that area of memory.

4. The commands are slower than standard Applesoft, but then again, they are dealing with twice as many pixels.

5. For those of a technical bent (meaning if you only write in BASIC you needn't read this), a single, previously unallocated, byte on zero page is used by these commands as a main memory/auxillary memory flag This was a necessary evil in order to maintain all of the standard entry points and fit the new features in to the system. The location used is $E3 and can be thought of as new addition to the internal and external cursors. It should not be changed by any assembly language commands that call the Double Hi-Res commands, and should not be tampered with between Double Hi-Res calls.

6. All of the standard Hi-Res errors are supported, but with the new bounds on color and horizontal location as described in Section 9.1.

7. The LNTYP= command is implemented by replacing the Lo-Res command COLOR= in the Applesoft command table. Thus if a program that uses the COLOR= command is loaded into memory while the Double Hi-Res commands are installed, all of the COLOR= commands will appear as LNTYP= commands. Likewise, all of the LNTYP= commands in a program will change to COLOR= commands if the Double Hi-Res commands are disconnected, as when the RESET is hit. No need to fear though for reinstallation of the commands will cause the program to revert back to its former form.

The only real problem arising from all of this is that if you are going to execute or enter into a program a LNTYP= command, then the Double Hi-Res commands must be installed. If they are not, the entry of LNTYP= will not be interpreted as a command but as an assignment statement.

Within a program, the best way to tell that the instruction LNTYP= has been taken as a command is to note if there is a space between the 'P' in LNTYP and the '='. Assignment statements place a space between the variable name and the '=' ('LNTYP=' is interpreted as a command, 'LNTYP =' is interpreted as an assignment). If LNTYP= is interpreted as an assignment, then you must either install the Double Hi-Res commands and retype the line or reenter the line as COLOR=, letting the Double HiRes commands convert it later.

>> DHR AMPERCHART.MOD <<

by Andy Scheck

FUNCTION: A full implementation of AMPERCHART, designed to be loaded and run using the DHR RELOCATE.MOD.TB and DHR JUMP FILE.TB modules that is to be used only on an Apple //e where Applesoft can be modified using the DHR MODIFIER.TB command.

LENGTH: 5858 bytes ($16E2)

SYNTAX: (Same as standard AMPERCHART.TB)

SAMPLE:
```
&"DHR AMPERCHART",GGO
 &"DHR AMPERCHART",PLT(UX,UY)
```

-> See Chapters 4 and 5 for a complete description of these and all of the rest of the AMPERCHART commands.

HOW TO USE IT: This version of AMPERCHART performs the same functions as DHR AMPERCHART. The only difference is that this version is not position independent (it can only function when installed at location $800), it requires Applesoft to be modified by DHR MODIFIER.TB to function properly. It is smaller than DHR AMPERCHART.MOD.TB so that it will safely fit under Hi-Res page 1. This version can only be used with DHR RELOCATE.MOD.TB.

To use this command, you have to perform the following steps:

1. The commands DHR RELOCATE.MOD.TB and DHR JUMP FILE.TB must be added to your program using the Workbench.

2. DHR AMPERCHART.MOD and DHR MODIFIER.TB must be present on the diskette in the active drive when your program is RUN.

In your program, be sure to use the RELOCATE command at the very beginning before any variables are defined (but after, of course, line #1 that activates the ampersand character!).

A program that uses DHR RELOCATE.MOD.TB and DHR JUMP FILE.TB might look like this:

```
 1 CALL PEEK(175) + 256 * PEEK(176) - 46
 2 & "RELOCATE"
10 &"CHART", GCO : REM AMPERCHART COMMAND
20 HCOLOR=15 : REM DHR COMMAND
```

LIMITATIONS: This command must be used in conjunction with DHR MODIFIER.TB and it must be loaded and run at location $800. It is therefore incompatible with the Apple //c. It is not Workbench compatible without the use of a jump file and should not be linked to a BASIC program using the Workbench. This command is not ProDOS compatible. Other limitations are described above.

>> DHR AMPERCHART.MOD.TB <<

by Andy Scheck

FUNCTION: A full implementation of AMPERCHART that is relocatable and compatible with the DHR MODIFIER.TB commands.

LENGTH: 6354 bytes ($18D2)

SYNTAX: (Same as standard AMPERCHART.TB)

SAMPLE:
```
&"CHART",GGO
 &"CHART",PLT(UX,UY)
```

=> See Chapters 3 and 4 for a complete description of these and all of the rest of the AMPERCHART commands.

HOW TO USE IT: This command is Workbench compatible and can be used almost exactly like the standard AMPERCHART.TB, once Applesoft has been modified to include the Double Hi-Res commands by DHR MODIFIER.TB.

There are only a few changes that the user must be aware of:

1. DHR MODIFIER.TB must be called before using this special version of AMPERCHART.

2. The limits on the horizontal screen coordinates (sxmin & sxmax) have been extended to 0-559.

3. Only a single page of Double Hi-Res graphics is supported so the GGO command has been modified, it no longer has an argument, and the WRK and DSP commands have been deleted. Thus &GGO will initialize AMPERCHART for Double Hi-Res plotting on page 1, but &GGO(1) or &GGO(2) will result in an error message. This was done intentionally to force you to be aware of which command you are using at any one time.

4. This version of AMPERCHART is larger than 6K bytes, thus eliminating the inverted installation option. This version of AMPERCHART must be either used with Routine Machine and the splitter command or it must be located above Hi-Res page 1. Again the reader is referred to the discussion of installation options in Chapter 7 for more details.

LIMITATIONS: Not compatible with the Apple //c.

The following pages contain further details on each of the menu choices presented when running the Workbench utility program. They are provided to answer any specific questions which might arise about a given menu choice.

10.1 MENU OPTION PREREQUISITES

For each of the options listed in the menu, certain conditions must be satisfied before the option can be selected. The prerequisites for each of the options are stated below:

1	ADD A COMMAND	There must be a program in memory.
2	REMOVE A COMMAND	At least one command must be added.
3	REMOVE ALL COMMANDS	At least one command must be added.
4	COPY ALL ADDED COMMANDS TO DISK	There must be added commands present.
5	RESTORE COMMANDS FROM DISK	This option can be exercised ONLY if no commands are present.
6	REPORT COMMANDS ADDED	At least one command must be present.
7	SEARCH FOR COMMANDS	There must be a program in memory.
8	DISPLAY MEMORY MAP	There are no prerequisites.
0	EXIT	There are no prerequisites.

10.2 OPTION #1: ADDING COMMANDS

Although most of the important points regarding this operation have already been covered in the beginning of this manual, one more comment is in order regarding this option.

When making up your name for an added command, the name used to call the command can be anything you like, subject to the following restrictions:

a) The maximum length of the command name is fifteen characters.

b) Control characters are not permitted; otherwise all keyboard characters other than ']' and quote marks (") are permitted.

c) The first character must not be a space; otherwise spaces within the name are permitted.

The name used for a command is also independent of the name of the Toolbox File used to add the command.

If you should forget the name you gave a particular command, you can always use Option #6 (Command Report) to get a report on the command names (and the original file names) of all currently added commands.

10.3 OPTION 2: REMOVE A COMMAND

Suppose you have added 23 commands and then decide you really don't want command #5. The Workbench allows you to remove it easily. Select '2' at the menu, and the first page of a command report will appear. This chart will show all of the added commands, giving both the command name, and the name of the Toolbox File from which the command was derived.

The commands are shown eight at a time. If you want to go to the next eight, simply press the space bar to advance. You can also return to the main menu at any point by pressing RETURN alone.

If you see the command you wish to remove, press the 'R' key to tell the program you wish to remove an item. Then enter the number of the command you wish to remove.

The Workbench will then remove the specified command and return you to the main menu.

OPTION 3: REMOVE ALL COMMANDS

It is also possible to remove all added commands at one time. Select option #3 for this function. You will be asked to confirm your intentions for such a drastic action. You must respond with either 'Y' or 'N' to this question.

SPECIAL NOTE: If you use Apple's Renumber program, or use use Hi-Res graphics with a program that is too long, you may destroy some of the added Toolbox commands. If this happens, item #3 in the main menu will change to 'REMOVE APPENDED MACHINE CODE', meaning that the Workbench can no longer recognize the added Toolbox commands. See the MEMORY MAP section later and Appendix A (A Little More Detail) for more information about this situation.

10.4 OPTION #4: COPYING ALL COMMANDS TO DISK

After using The Workbench for a while, you will probably find that there is a certain group of commands which you almost always put in your programs. Option 4 may be used for saving an assembled group of commands to disk as a single file. This enables you to add the entire group of commands at one time (using Option 5) to an Applesoft program in the course of development.

Such a mini-library can really come in handy when you write a variety of different programs. An example of a group of commands which could make up a good mini-library include:

a) AMPERSAND RESTORE.TB
b) RESET ONERR.TB
c) STRING INPUT.TB
d) ERR.TB

To save a mini-library to disk, select item #4 in the main menu of the Workbench.

The screen will then display:

```
ALL ADDED COMMANDS WILL NOW BE
COPIED TO A FILE ON DISK

DO YOU WISH TO USE 'CMD FILE'
AS THE FILE NAME? [Y]
```

If you press 'Y' (or just RETURN) here, the Workbench will save a copy of all the added commands to the diskette under the name 'CMD FILE'.

If you want to give the file a different name, just press 'N' and enter the name of your choice. This is the recommended method, so that you don't accidentally overwrite another previously saved mini-library.

The main purpose for the 'CMD FILE' name option is for temporary saving of a group of commands. This is useful when using the second main application of this option, which is to save added commands to disk to enable the performance of some operation which might damage appended machine code, i.e. added Toolbox commands.

Although none of Roger Wagner Publishing's software will damage added commands, the same cannot be said for all utilities. In particular, APPLE COMPUTER'S RENUMBER program WIPES OUT all commands (or machine code) added to an Applesoft program.

It is not difficult to use Apple Computer's Renumber program to renumber an Applesoft program which has Toolbox Files added, but to do so you must first use Option #6 to copy all added commands to disk.

Then use Option #3 to remove all added commands. You may now renumber your program (or use whatever other utility you wish here). Finally, use Option #5 to restore the commands that you saved to disk with Option #6. This procedure is not required for any utility that does not destroy appended machine language at the end of an Applesoft program.

10.5 OPTION #5: RESTORING ADDED COMMANDS FROM DISK

If you have used option #3 to remove added commands your program, or if you are just starting a new program, to which you wish to add a pre-assembled mini-library, you may use Option #5 (RESTORE ADDED COMMANDS FROM DISK) by following the procedures listed here:

1) If the Workbench is in memory, re-enter with a CALL 2051. If the Workbench is not in memory, enter BRUN WORKBENCH.

2) When the menu appears, select Option #5 by pressing '5'. You will be asked to confirm by pressing 'Y' (for Yes) or RETURN alone.

) The program will then display:

IS THE COMMAND FILE TO BE RESTORED
CONTAINED IN DISK FILE 'CMD FILE'? [Y]

If you have not used this file name to save the added commands, enter 'N' (for NO). It will then display:

WHICH FILE NAME THEN? (<RET> = QUIT)
(FOR CATALOG ENTER 'CAT')

You may now either catalog the disk or enter the name under which the commands were saved.

4) The Toolbox will now restore the commands which were formerly copied to disk, and return you to the menu.

If the added commands were originally copied to a disk file under the name 'CMD FILE', then the Workbench will delete this file from the disk. If you specified some other name for the disk file, then it will not be deleted.

10.6 OPTION #6: REPORT COMMANDS ADDED

The Workbench may be used to add up to 255 commands to an Applesoft program. Although it is unlikely you will even approach this number, you may from time to time wish to see a list of the commands which have been added to a particular Applesoft program.

Pressing '6' at the main menu will produce a Command Report. This is useful because it allows you to see what commands have already been added and to check on the names given to them.

When selecting each of the 'report' options 6 - 8, you are asked if output is to go to the printer. If you simply want screen display, press RETURN (or 'N'). If you want a print-out, press 'Y'. The Workbench will then ask:

PRINTER IN SLOT: 1

If this is the correct slot for your printer, press RETURN, otherwise enter the slot your printer is in.

For each command added you are informed of the command name and the name of its disk file.

The Command Report displays [illegible]
If you have more than eight commands added [illegible] space bar to go to each successive page (or you can return to the menu at any point by pressing RETURN).

It is also possible to transfer directly from the Command Report to the Search option by entering 'S'.

10.7 OPTION #7: SEARCHING FOR TOOLBOX COMMANDS

This option is provided so that you can list every line in your program that uses a Toolbox command. This is especially useful when combined with the Command Report option.

The search function has three options: (1) all statements in your program using an ampersand (or CALL), (2) all Toolbox commands, or (3) use of only a particular command.

To see how the SEARCH option works, follow these examples:

1) From the Workbench, select Option #0 to exit the program. Then type in 'NEW' to clear memory.

2) Now enter LOAD BINARY FILE COPIER (on the back of the Wizard's Toolbox disk).

3) When the Applesoft prompt returns, enter CALL 2051 to re-enter the Workbench.

4) At the menu, select Option #7 and press RETURN. The Workbench will then display the following:

```
SEARCH TYPE:   AMPERSAND STATEMENTS

DO YOU WISH TO SEARCH FOR:

1  ALL SUCH STATEMENTS?
2  ALL TOOLBOX COMMANDS?
3  A PARTICULAR COMMAND?

(PRESS 'T' TO CHANGE SEARCH TYPE,
OR <RETURN> TO QUIT)
```

5) Select Option #1 to display all of the ampersand statements. Since BINARY FILE COPIER contains a considerable number of ampersand statements it will be necessary to press the space bar once or twice to display them all (remember only eight are displayed at a time). Notice that on each page the Workbench displays:

PRESS "S" FOR NEW SEARCH
OR <SPACE> TO CONTINUE

at the bottom of the screen to prompt you for the next page of ampersand statements, as well as to provide you the option of changing the search to CALLS instead of AMPERSANDS.

Since the ampersand is used in the program only to call Toolbox commands, and this is the only method used to call the commands, you will see the same display whether you select option 1 or option 2.

Normally commands will use the ampersand, but there is an alternative method which employs a CALL directly to the Toolbox commands themselves. (See Appendix B for details.) Thus when searching for the Toolbox commands in your program you may specify that the search is to be for ampersands or for CALLs. (The latter option also allows you to search for all CALLs in an Applesoft program whether or not they are Toolbox commands.) You can toggle between these two options by pressing "T".

To see how this works, continue as described here:

6) Press "S" to return to the search menu, and press "T" to change to the search for CALLS and repeat the steps to display the CALLS used in the program. Notice the top line of the search menu now appears as:

SEARCH TYPE: CALL STATEMENTS

When an ampersand or CALL statement is found which satisfies the conditions of the search, it is displayed along with the number of the line in which it occurs.

If the line consists of more than one statement, as in the statement below:

```
N = INT(VAL(LEFT$(B$(J),4))): &"TONE": FOR I=1 TO N:
PRINT A$(I);: PRINT " = "A(X(I)): PRINT: NEXT
```

then only the command statement itself will be displayed, as in:

&"TONE"

SPECIAL NOTES:

If you are using the CALL method of using Toolbox commands, a typical command statement would be:

CALL IR "NAME", A$, B$

You might erroneously omit the 'IR' so that your program would contain the statement:

CALL "NAME", A$, B$

This would make a nice method of using a command, except that it doesn't work! "NAME" (a string literal) cannot be evaluated as a calling address.

If you use the SEARCH OPTION, and the Workbench finds a CALL statement with some expression following the CALL which cannot be evaluated as an address, it will display the offending statement along with an error message, and return you to the immediate mode of Applesoft. This gives you the opportunity to correct the CALL (by replacing, e.g., CALL "NAME" with CALL IR "NAME"). You can then re-enter the Workbench with the usual CALL 2051 and resume normal operations.

10.8 OPTION #8: THE MEMORY MAP

The use of this option is not required for adding or removing commands, but does provide useful information such as the number of bytes that the added Toolbox commands are adding to an Applesoft program. In addition, use of this function is highly recommended if you are using Hi-Res graphics.

The Memory Map is selected by choosing Option #8 from the main menu of the Workbench. As with previous 'report' options, you may output the Memory Map to the printer by entering a 'Y' when the Workbench asks you about printer output.

As noted before, the Workbench occupies memory from $800 to $25FF, and when it is present your Applesoft program starts at $2601 instead of the usual location of $801 (see the diagrams on pp.84-85 of this manual).

Normally you will be more interested in the location of your program at run time, rather than its location when the Workbench is present.

Thus when the Memory Map is first displayed it shows the location of your program AS IF it were at $801. If you then press 'A' the Memory Map will change to display the current actual addresses (with your program at $2601). You can toggle back and forth between these two modes using the 'A' key.

The Memory Map does not always have the same format. It differs according to whether there is a program in memory and whether it has any added commands.

When there is no program in memory, the Memory Map displays only four addresses, such as the following:

```
PROGRAM START:          $0801 =  2049
LOMEM:                  $0804 =  2052
HIMEM:                  $9600 = 38400
DOS BUFFERS:            $9600 = 38400
```

Considerably more information is displayed when there is a program in memory. The following is a typical Memory Map, corresponding to a situation like that shown in Figure 3 on page 85 of this manual.

```
                APPLE ][ MEMORY MAP

PROGRAM START:                  $0801 =  2049
BASIC PROGRAM END:              $0F5F =  3935
ACTUAL PROGRAM END:             $1289 =  4745
LOMEM = SIMPLE VARIABLES:       $1289 =  4745
ARRAY SPACE:                    $12BA =  4794
ARRAY SPACE ENDS:               $12E2 =  4834

STRINGS:                        $936E = 37742
HIMEM = END STRINGS:            $9600 = 38400
DOS BUFFERS:                    $9600 = 38400

BASIC PROGRAM LENGTH:           $075E =  1886
TOOLBOX BYTES ADDED:            $032A =   810
TOTAL PROGRAM LENGTH:           $0A88 =  2696
FREE SPACE:                     $808C = 32908
STRING STORAGE:                 $0292 =   458
NOTHING UNDER DOS BUFFERS

MAXFILES = 3
```

The most interesting thing to notice in this chart is the entry "TOOLBOX BYTES ADDED" compared to the entry "BASIC PROGRAM LENGTH". In the above example, the program is 1886 bytes long, while there are 810 bytes of machine code appended in the way of Toolbox commands. This means that this program is approximately 30% machine code. This percentage is often even greater as you use more and more Toolbox commands.

The net result is that you'll find that even though it appears you are writing a program in BASIC, it is not unusual to find that over 50% of the final program is written in fast and compact machine code!

HI-RES GRAPHICS AND THE MEMORY MAP

The most important function of the memory map is if you are using Hi-Res graphics in your program. Because the Hi-Res page occupies the middle memory range of your computer, an Applesoft program can become so long as run into the area used for graphics. The symptoms of this are program crashes after an HGR or HGR2, or strangely changing variable values while using graphics.

The memory map can be used to predict when a problem is likely to occur.

1. If the ACTUAL PROGRAM END value is greater than $2000 ($4000 if you are using HGR2), then your program is too long.

Solution: Remove REMark statements from your program and take other steps to shorten the length of the listing, such as combining lines where possible. You can also use utilities such as Roger Wagner Publishing's APPLE-DOC II to compress your BASIC program, or use the Chart 'n Graph Toolbox to move your program above the Hi-Res page.

You can also add the following line to your program:

```
2   IF PEEK(104)=8 THEN POKE 104,64:POKE 16384,0:PRINT
CHR$(4);"RUN MYPROGRAM": REM FOR HGR USE
```

or:

```
2   IF PEEK(104)=8 THEN POKE 104,96:POKE 24576,0:PRINT
CHR$(4);"RUN MYPROGRAM": REM FOR HGR2 USE
```

where 'MYPROGRAM' is the name of your program as stored on the diskette.

If your program does not exceed $2000 (or $4000 for HGR2), then you should also remember to always use a LOMEM: 16384 (24576 for HGR2) as line 2 of your program.

If you are unfamiliar with the other memory locations mentioned in the memory map, we recommend "ALL ABOUT APPLESOFT", available from local computer stores and also from Roger Wagner Publishing.

APPENDIX A

A LITTLE MORE DETAIL

The information in this section gets a bit technical. It is included here for the enjoyment of the programmer that wants to know more about how the Toolbox actually operates.

Although it is not necessary to understand all the information provided here to use Toolbox system, it may help you get a better feeling for what you are actually doing when you use it. It is also a required background for the information that follows in the next sections on writing your own Toolbox files.

In addition, there is a description of the conflict between a long Applesoft program and the Hi-Res pages, which may be helpful if you are using graphics.

When you first specify the name of a command to be put in your Applesoft program, the Workbench does the following things:

1) It appends the Toolbox File to the end of your Applesoft program. Because of the nature of Applesoft, this machine code is not visible during a LIST, and is unaffected by adding or deleting normal BASIC lines or statements.

2) It saves both the name you give to use the routine, and the name of the disk file loaded. This is to make later identification of commands easier (see the section on the Command Report).

3) The first time a command is added, the Workbench adds its own 'Interface Routine' to the end of your program. The function of this is to locate a routine named in a Toolbox command and to pass control to it.

The manner in which memory is normally allocated in the Apple computer is as follows:

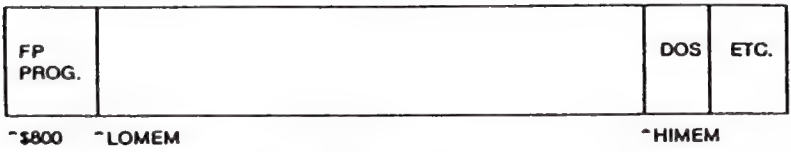


Figure 1: "Normal" Applesoft program

Normally an Applesoft program resides at the "bottom" of memory, with all remaining space above it reserved by DOS and for variable storage.

The Workbench could be loaded right after your program, but as Toolbox Files were added to the end of the Applesoft program, they would start to conflict with Workbench itself. To avoid this, the Workbench moves your program UP in memory, and locates itself at memory location $800 (the dollar sign indicates a hexadecimal address).

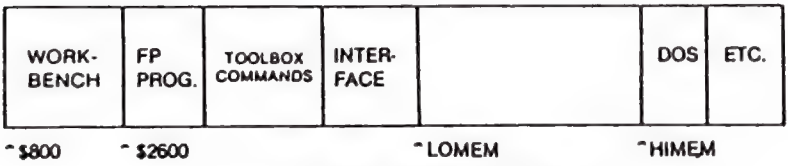


Figure 2: With the Workbench Installed

With your program safely relocated to $2600, routines are automatically placed between the Interface Routine and the end of your program ('FP PROG' on the chart). When your program is run, the Line #1 that was put in by the AMPERSAND SETUP file points the ampersand vector to the Interface Routine at the end of your program. From there the name following the ampersand is read and the proper routine used.

CAUTION: Note that your program now STARTS at $2600. This is well into the Hi-Res page 1 display, which normally starts at $2000. Thus, any program using an HGR command will destroy itself if you attempt to run it with

the WORKBENCH in memory. The Workbench should be removed before running any program using Hi-Res graphics.

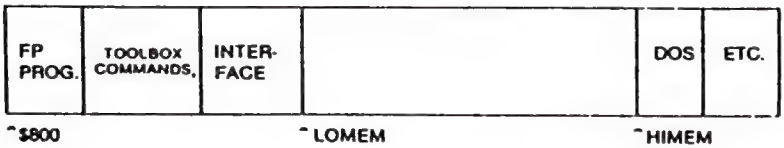


Figure 3: At 'Run Time'

If you use the REMOVE WORKBENCH utility, or do a fresh load of an Applesoft program containing appended Toolbox files after entering 'FP', then memory will be allocated as in Figure 3. This shows the Applesoft program back at its normal location, with the appended routines and the Interface Routine still present.

Hi-Res graphics on page 1 will in most cases now be available. When in doubt, you can use the Memory Map option from the Workbench (see p.78 of this manual) showing the addresses for your program at its normal location of $801.

When developing a program using Toolbox commands, you can proceed in three ways:

1) Add commands to your program, one at a time, as required.

2) Work on the BASIC portion of your program first, and then add the required commands later, or

3) Add all required commands and then write the BASIC portion of the program which calls them.

HI-RES GRAPHICS, RENUMBERING AND OTHER HAZARDS TO THE TOOLBOX

Improper use of Hi-Res graphics, or use of Apple's Renumber program can destroy added Toolbox commands in your Applesoft programs.

If you are using Hi-Res graphics with the Toolbox system, you should be sure to read the section on the Workbench's MEMORY MAP option. This will help you avoid

having an HGR or HGR2 command destroy added Toolbox commands.

If you are using Apple's Renumber program, be sure to read the section in this manual on COPYING ALL COMMANDS TO DISK in the Workbench. (is this there?)

If the worst should happen and your added Toolbox commands are damaged, you will be able to tell because the Workbench main menu will say NO COMMANDS ADDED, and option #3 will now say REMOVE APPENDED MACHINE CODE.

This is because the Workbench can tell that you have something (destroyed Toolbox commands probably) still added to your program, but it doesn't recognize it as the Toolbox commands.

This can be verified by going to the MEMORY MAP, where the screen will now display BYTES APPENDED where it used to read TOOLBOX BYTES ADDED. Again, the Workbench is telling you that something is left hanging on the end of your program, but it doesn't know what.

If this should happen, you will have to use option #3 to remove the damaged Toolbox commands, and then individually re-add the various Toolbox commands used by your program. Option #7 may be helpful here as it can be used to list all the Toolbox commands called by your program.

APPENDIX B

ADVANCED USE OF THE TOOLBOX SYSTEM

1. USING CALL STATEMENTS IN TOOLBOX COMMANDS

Normally, the pointer that Applesoft uses to indicate the end of the Applesoft program in memory does in fact point at the end of the last line of a program.

However, it is possible to redirect this pointer so that it points to a location some distance after the end of the BASIC lines in a program. This in effect creates a "pocket" at the end of a program into which may be placed machine language programs.

The beauty of this approach is that when the program is loaded or saved to disk, or even when lines are changed, added or deleted from the program, the pocket within the entire Applesoft program "envelope" (i.e. the portion of memory between the start-of-program and end-of-program pointers) is protected and carried along with the BASIC program.

Once a routine has been appended in this way, the question arises as to how it is to be called by the Applesoft program. If the routine is to be appended to a finished Applesoft program, and that program is always run at the same address in memory, then you could conceivably CALL the routine at a fixed address.

This method, though, is not very practical, since programs are often changed, with the result that the absolute memory locations of any appended routines change accordingly.

Fortunately, there is an alternative. If one machine language routine of length N bytes is appended to the end of an Applesoft program, then the address of the first byte of the routine will always be found at the address of the end of the program minus N. (The term 'program' here means BOTH the BASIC portion together with the appended machine code.) That is to say, one would look at the value contained in PRGEND (the pointer to the end of program: 175,176) to determine the end of the program 'envelope', and then subtract N bytes.

Thus the routine could be called at the address:

```
PEEK(175) + 256 * PEEK(176) - N
```

This address will remain valid despite any changes in the length of the BASIC portion of the program, since the pointer at locations 175,176 (PRGEND) always points to the end of the program envelope.

This method can be generalized to accommodate more than one appended routine, but to append several routines by hand can be rather tedious. Fortunately, you don't have to, since the Workbench will do it all for you!

Whenever you have added routines to your program using the Workbench, there will also be present a machine language routine which handles the initial part of your Toolbox command. This routine is automatically included in your program when you add the first command.

This routine is known as the 'Interface Routine' because it acts as an interface between your Applesoft program and its appended machine language routines.

It is always placed at the very end of the program envelope, and occupies the last 203 bytes. Thus it can be called at the address:

```
IR = PEEK(175) + 256 * PEEK(176) - 203
```

This is the address set up in line #1 of your program by the EXEC file on the Database Toolbox diskette named CALL SETUP.

When, during the course of the execution of your Applesoft program a Toolbox command is called (by one means or another), the first thing that happens is that control passes to the Interface Routine, which then looks at the command name (between the quotes) in an attempt to find out which particular command you're interested in.

With the name in hand it then searches through the commands added, and if it finds the one you want then the Interface Routine passes control on to that routine.

2. USING THE AMPERSAND IN TOOLBOX COMMANDS

Although the routines can be called using the CALL statement, there is a more efficient method, using the ampersand.

At locations $3F5-3F7 (decimal 1013-1015) is a JMP instruction (see the Apple][Reference Manual, p. 132). If the address of the Interface Routine is placed into

locations 1014 (low byte) and 1015 (high byte), then when the Applesoft interpreter encounters the ampersand character (&), control will pass to the Interface Routine. As before, the routine will then locate the command that you wish to use, and pass control to it.

Thus in this system there are TWO distinct ways of calling added commands. The effects are the same, and only the syntax of the Toolbox command varies. Below are examples of commands using each of the two methods:

```
CALL IR "SWAP", A$, B$
     & "SWAP", A$, B$

CALL IR "SORT", A$(FE) TO A$(LE)
     & "SORT", A$(FE) TO A$(LE)
```

The two methods have the same result: To call the Interface Routine, which then reads the name of the command, locates it in memory (somewhere between itself and the end of the BASIC portion of your program), and then JMPs to the routine.

If the CALL method is used, it is necessary to have a statement in your Applesoft program which defines the CALL address 'IR'. Such a statement can be inserted in your program (as line #1) by EXECing the file CALL SETUP on the Database Toolbox diskette (with your program in memory, of course).

If the ampersand method is used, then the ampersand vector must be set up to point to the address of the Interface Routine. This can be done by a CALL to an internal entry point in the Interface Routine itself.

This internal entry point is 46 bytes before the end of the Interface Routine, which is itself at the end of your Applesoft program, and so the required ampersand vector setup can be accomplished with a simple

```
CALL PEEK(175) + 256 * PEEK(176) - 46
```

The file AMPERSAND SETUP on the Database Toolbox diskette can be EXECed to insert this line (as line #1) in your program.

Then when your program is run the ampersand hook-up is automatically made and your program can then use the ampersand for added commands without further thought.

It is up to you which method you wish to use to call commands. The ampersand method is more pleasing to the eye, and also involves fewer keystrokes, but it does modify the ampersand vector which may be used by other utilities. If you wish to use ampersand utilities, you may care to use the CALL IR method of calling Toolbox commands. It is also possible that you might have an ampersand routine that is entirely independent of the Toolbox system. In that case, the CALL IR method will leave the ampersand free for other duties.

If you are still unsure as to the difference between these two methods of calling commands, then simply ignore the existence of the CALL method, and stick to ampersands (they'll never let you down).

3. RESTORE AMPERSAND.TB

A possible conflict can arise between the use of the ampersand to call Toolbox commands and its use with utilities such as Apple Computer's Renumber routine.

If the ampersand has been set to "point" to one particular utility, such as a Renumber Program, then when a program using a Toolbox command runs, this pointer is lost because the Toolbox system takes over the use of the ampersand.

There are two solutions to this problem. The first is to call commands using a CALL directly to the Interface Routine as described in the previous section on CALLs.

If this method is used then the ampersand vector is not reset when your program is run (unless your program resets it in some other way), and so your ampersand-related utility is always connected.

However, normally the method of calling commands by means of the ampersand is preferable. In this case we may employ a Toolbox command to solve the problem. When your program is run, the line which causes the ampersand vector to be reset to the Interface Routine also causes the old ampersand vector to be saved.

Included on your Database Toolbox diskette is the routine RESTORE AMPERSAND.TB. This can be added to your program like you would any other command. It should then be used just before your program ends; it has the effect of restoring the original ampersand vector. Thus on

return to immediate command mode your ampersand-related utility will have been reconnected and will be ready to be used.

If the RESTORE AMPERSAND.TB routine is used, it MUST be the LAST Toolbox command used before your program comes to an end. If not then a subsequent Toolbox command will have the effect of calling your utility from within your program, with unpredictable consequences.

4. Using JUMP FILE CREATE

There is a wealth of relocatable routines available for performing all kinds of tasks in the other Toolbox library disks. Thus normally you will not have to be concerned with interfacing any non-relocatable routine with an Applesoft program. However, it may be that you already have a routine which you have been using at a fixed address and which is not relocatable. Such a routine cannot be used directly as a Toolbox command. There are, however, two possible solutions.

The first is to take your routine and make it relocatable. If this cannot easily be done then there is a utility on the Database Toolbox diskette called JUMP FILE CREATE to provide aid and solace.

Using this utility you can create an intermediate calling routine which will function as a link between the Interface Routine (appended to your Applesoft program) and your non-relocatable routine (occupying its required position in memory).

The only requirement for using JUMP FILE CREATE is that you know the address of the routine which would normally be CALLed (or used via an ampersand).

For example, let's suppose that the file TONE.TB were a nonrelocatable routine which MUST be loaded at location $300 (decimal 768) to function properly. To create an interface command, run JUMP FILE CREATE.

You will then be presented with the opportunity to enter the call address in decimal. If you do not know the decimal address, press RETURN and a hex option will be presented. For our example, enter '300' as the hex address. It will then print out the decimal equivalent as a matter of information, and instruct you to press 'S' to save the file.

It will then be saved to the Toolbox diskette as "JMP.$0300/768.TB". This Toolbox File could then be added to your program like any other Toolbox File, using the Workbench. If when adding the JMP file, you specify "TONE" as the command name, then your program could use TONE.TB as follows :

```
1 CALL PEEK(175) + 256 * PEEK(176) - 46
10 PRINT CHR$(4);"BLOAD TONE.TB, A$300"
20 INPUT "PITCH,DURATION?  "; P, D
30 & "TONE", P, D
```

If you wish to write your own machine language routines, the Toolbox system is an ideal way to interface them to an Applesoft program.

In addition, the following aids to machine language programming are available through Roger Wagner Publishing:

* MERLIN, an extremely powerful, yet easy to use macro assembler. Besides being one of the best assemblers currently available for the Apple, MERLIN offers additional features like Sourceror, which disassembles raw binary code into pseudo source files, and a fully labeled and commented source listing of Applesoft BASIC. MERLIN also uses the upper 16K of memory to become a co-resident assembler with the ability to easily switch between MERLIN and Applesoft.

* MERLIN PRO, the expanded version of Merlin for both DOS 3.3 and ProDOS. Has all the features of Merlin, plus a local labels, and a relocating linker and loader, plus use of the full 128K of RAM on an Apple //e or //c (required).

* MUNCH-A-BUG, an excellent de-bugging package for machine language programs. Allows use of labels and includes its own mini-assembler. Best of all, it can be set in a "waiting" condition to go into the trace mode only after a given routine has been called, thus allowing the user to trace machine code from within fully operational Applesoft programs.

* ASSEMBLY LINES: THE BOOK, is a beginning tutorial on machine language programming. There is a section on creating relocatable code, and also other topics such as sound generation and disk I/O.

* APPLESOFT IN DEPTH, is an extensive study of Applesoft and related routines. Published by CALL -A.P.P.L.E., and also available from Roger Wagner Publishing, this is an invaluable reference book for programmers.

On the other hand, if you are unable or not inclined to delve into what may seem to you the Terra Incognita of assembly language programming, yet find yourself in need of a command to do a certain task, then please contact Roger Wagner Publishing.

Although we do not usually produce custom Toolbox commands, requests of general interest do often result in new Toolbox commands, and we'll be sure to let you know if a command that interests you becomes available.

```
**************************************************************
*  IMPORTANT:  IF  YOU  DO  WRITE  YOUR OWN ROUTINES,        *
*  PLEASE WRITE  ROGER  WAGNER PUBLISHING!  WE MAY BE        *
*  ABLE TO  INCLUDE  THE  ROUTINE  ON FUTURE  TOOLBOX        *
*  COMMAND LIBRARY DISKS!  WE CAN ALSO SUPPLY INFOR-         *
*  MATION  AS TO  WHAT  ROUTINES ARE  CURRENTLY UNDER        *
*  DEVELOPMENT, AND WHICH ARE STILL NEEDED.                  *
*                                                            *
**************************************************************
```

ADVANCED TOPICS

This section covers some of the not so obvious but potentially useful techniques for getting the most out of AMPERCHART. Some of these were touched on in the TUTORIAL section and others are acually implemented in the demos on the AMPERCHART disk. No specific order was chosen for presenting these (other than the order that the authors used them).

Before introducing these techniques, we will present a brief description of what the AMPERCHART work area is really like. It was mentioned earlier that AMPERCHART lets you plot outside the defined clipping window without generating any errors. That is true to a point. AMPERCHART will function without generating errors as long as no points are outside the limits of -32767 to +32767 in modified Applesoft screen coordinates. This allows plotting to a "virtual" screen that is about 234 times as large as the visible screen in the horizontal direction and 343 times as large in the vertical direction.

It won't happen very often, but when "blowing up" plots (too much) to look at small detail, these limits may get exceeded causing an "ILLEGAL QUANTITY ERROR". It should be pointed out that pushing these limits out past the +/-32000 range may cause inaccuracies. Remember, these limits are only in the virtual screen; the user coordinates, for purposes of scaling, are limited only by the floating point number range of Applesoft.

Version Number

Accessing the following internal locations of AMPERCHART is accomplished by using offsets from the begin-..ing of AMPERCHART. AMPERCHART's starting location is always available through the LOC function. The offsets listed are constant for a given version of AMPERCHART, but since there are three standard and two DHR versions included to suit your particular needs, there are five sets of offsets. For your program to be able to run, independent of which version of AMPERCHART is in memory, the particular offsets can be chosen based on the version number. This number is always found 25 bytes beyond the value returned by LOC.

Although there are numerous entry points and global storage locations that have been designed into AMPERCHART for any future additions or machine language interfaces, these are by far the most useful: OLDX, OLDY, OLDC, and RESET.

The storage locations OLDX and OLDY are somewhat equivalent to the internal graphics cursor of Applesoft. They hold the location of the last point addressed in the virtual graphics screen's coordinates. Doing a MOV and then a couple of PEEKS into these locations provides the inverse of the STU function. OLDC holds the position code of the last point addressed a value of 0 means the point was inside the clipping window. Values other than 0 indicate whether the point was above, below, left of, or to the right of the clipping window (determining the particular bit settings is an exercise left up to the user if he should need to know that information).

Sometimes it is useful to be able to reinitialize AMPERCHART without clearing the screen or changing the screen switches. This is done by CALLing the RESET entry point. This is equivalent to doing the following:

```
&CLP(0,279,0,191): &SCL(0,279,0,191)
&TSZ(2,3): HCOLOR=3.
```

The locations of these internals for each of the five AMPERCHART versions are given in the following five tables:

AMPERCHART.PLOT.TB - BASIC PLOTTING
INTERNAL PROGRAM LOCATIONS OF INTEREST

[All locations are offsets from LOC(X)]

Name	Description	Offset Dec	(Hex)
VERSN#	Version # (1-3, =1 for this version)	25	($ 19)
RESET	Reset default parameters without disturbing display	1597	($ 63D)
OLDX	X value of last point plotted	2515,6	($ 9D3,4)
OLDY	Y value of last point plotted	2517,8	($ 9D5,6)
OLDC	If this location=0 then last point was in window	2519	($ 9D7)

NOTE: Locations $9D3-$9D6 can be used to perform the inverse function of STU.

AMPERCHART.PLOT+LABEL.TB - BASIC & LABELING
INTERNAL PROGRAM LOCATIONS OF INTEREST

[All locations are offsets from LOC(X)]

Name	Description	Offset Dec	Offset (Hex)
VERSN#	Version # (1-3, =2 for this version)	25	($ 19)
RESET	Reset default parameters without disturbing display	2148	($ 864)
OLDX	X value of last point plotted	3082,3	($ C0A,B)
OLDY	Y value of last point plotted	3084,5	($ C0C,D)
OLDC	If this location=0 then last point was in window	3086	($ C0E)
HLBROT	HLB rotation value	1041	($ 411)
VLBROT	VLB rotation value	1215	($ 4BF)
SCLSTR	HLB & VLB scale value	922	($ 39A)
DRWSET	1 for DRAW, 93 ($5D) for XDRAW of labels	928	($ 3A0)
TABLE	Built in shape table location	3101	($ C1D)

NOTE: Locations $C0A-$C0D can be used to perform the inverse function of STU.

AMPERCHART.TB - COMPLETE SYSTEM

Name	Description	Offset Dec	Offset (Hex)
VERSN#	Version # (1-3, =3 for this version)	25	($ 19)
RESET	Reset default parameters without disturbing display	3961	($ F79)
OLDX	X value of last point plotted	4915,6	($1333,4)
OLDY	Y value of last point plotted	4917,8	($1335,6)
OLDC	If this location=0 then last point was in window	4919	($1337)
HLBROT	HLB rotation value	2854	($ B26)
VLBROT	VLB rotation value	3028	($ BD4)
SCLSTR	HLB & VLB scale value	2735	($ AAF)
DRWSET	1 for DRAW, 93 ($5D) for XDRAW of labels	2741	($ AB5)
TABLE	Built in shape table location	4935	($1347)

NOTE: Locations $1333-$1336 can be used to perform the inverse function of STU.

DHR AMPERCHART.TB
and DHR AMPERCHART.MOD.TB

INTERNAL PROGRAM LOCATIONS OF INTEREST
[All locations are offsets from LOC(X)]

Name	Description	Offset Dec	Offset (Hex)
VERSN#	Version # (-19 for this version)	25	($ 19)
RESET	Reset default parameters without disturbing display	4082	($ FF2)
OLDX	X value of last point plotted	5133,4	($140D,E)
OLDY	Y value of last point plotted	5135,6	($140F,10
OLDC	If this location=0 then last point was in window	5137	($1411)
HLBROT	HLB rotation value	2886	($ B46)
VLBROT	VLB rotation value	3060	($ BF4)
SCLSTR	HLB & VLB scale value	2767	($ ACF)
DRWSET	1 for DRAW, 93 ($5D) for XDRAW of labels	2773	($ AD5)
TABLE	Built in shape table location	5153	($1421)

NOTE: Locations $140D-$1410 can be used to perform the inverse function of STU.

DHR AMPERCHART
and DHR AMPERCHART.MOD

Name	Description	Offset Dec	Offset (Hex)
VERSN#	Version # (=147 for DHR AC.MOD) (=179 for DHR AC)	25	($19)
RESET	Reset default parameters without disturbing display	3782	($ EC6)
OLDX	X value of last point plotted	4833,4	($12E1,2)
OLDY	Y value of last point plotted	4835,6	($12E3,4)
OLDC	If this location=0 then last point was in window	4837	($12E5)
HLBROT	HLB rotation value	2586	($ A1A)
VLBROT	VLB rotation value	2760	($ AC8)
SCLSTR	HLB & VLB scale value	2467	($ 9A3)
DRWSET	1 for DRAW, 93 ($5D) for XDRAW of labels	2473	($ 9A9)
TABLE	Built in shape table location	4853	($12F5)

NOTE: Locations $12E1-$12E4 can be used to perform the inverse function of STU.

Graphics Flashing Cursor

The RVS function can be useful for displaying a flashing cursor in graphics mode. A single character location is 7 dots wide and 8 dots high. To place a cursor at location X,Y (screen coordinates) on the screen is as easy as executing the function &CLP(X-3, X+3,Y-3,Y+4). Then a couple of &RVS's with an appropriate delay between them will cause that clipping window to "flash" without destroying the contents of that mini-clipping window. Needless to say, almost any size cursor may be simulated this way. One that can't is the single line. A clipping window must be at least 2 dots wide and high. To get a flashing underline cursor, for example, try using the HLB function with the XDRAW mode set (see USING ALTERNATE SHAPE TABLES below).

Clearing to a Color

While reading the section on CLR (of course you are reading the advanced stuff last) you may have thought that it would be nice to clear the clipping window to some color other than BLACK1. You may have already come up with the answer to that one. After clearing, set HCOLOR=your color, then use FIL. That's not all that inconvenient is it?

Multiple Screens - Super Hi-Res on an Apple][+

This is most useful for those of you with printers that allow bit-mapped graphics dumps, but don't own an Apple //e and extended 80 column card. Since AMPERCHART lets you draw all day long outside the screen limits, you can get higher resolution by setting the scale values so that the left half of your picture is drawn on Hi-Res page 1 and the right half on Hi-Res page 2.

There are many printer drivers that will dump page 1 and page 2 side by side. The only tricky part is getting labels placed correctly, as characters are not displayed unless the whole character will fit. I've used this technique to get as many as three full screen dumps placed horizontally across the paper, (you do know about Hi-Res page 3 don't you? Just do a "POKE 230,96 : &CLR" and draw to your heart's content. There are a few hitches like you can't see what you're doing and you aren't left with much program space, but you can get used to it.) that, combined with also splitting up the picture vertically can give some "Super Hi-Resolution

Plots"; of course you will have to write your own printer drivers to take full advantage of that.

Double Hi-Res Routine Entry Points

DHR MODIFIER.TB modifies Applesoft in such a way that all of the standard internal graphics entry points are preserved! This means that once the Double Hi-Res routines are installed, most Applesoft graphics programs, even those using assembly language utilities, will still work.

The only exceptions are those utilities that manipulate the graphics screen directly, without using the Applesoft graphic internals or those that use unpublicized entry points. . (AMPERCHART, for reasons of speed, falls into this former category and special versions of AMPERCHART are required to utilize the Double Hi-Res mode.) The following entry points are maintained in the modified Applesoft:

HGR	$F3E2	INCRX	$F48A	HFNS	$F689
SETHPG	$F3EA	INTY	$F4D3	HCOLOR	$F6E9
HCLR	$F3F2	DECRY	$F4D5	SETHCOL	$F6EC
BKGND	$F3F6	INCRY	$F504	GETHPLT	$F6FE
HPOSN	$F411	HLINRL	$F530	GETROT	$F721
HPLOT	$F457	HLINE	$F53A	GETSCAL	$F727
PLOT	$F45A	HFIND	$F5CB	GETDRAW	$F72D
INTX	$F465	DRAW	$F601		
DECRX	$F467	XDRAW	$F65D		

For a description of these routines and their calling requirements see the articles by John Crossley and C.K. Mesztenyi in Call-A.P.P.L.E. in Depth #1, All About Applesoft. This is available from both the Call - A.P.P.L.E. user group and Roger Wagner Publishing.

APPENDIX D

EASY CHART INSTRUCTIONS

EASY CHART is a demonstration program designed to show you how powerful AMPERCHART is. It is a complete menu-driven graphing program designed to fit the needs of a wide variety of users. It provides for data entry, editing and storage; plotting data in bar chart, pie chart or line chart form; and image editing such as labeling, filling, and erasing. The program is listable, modular, and commented so it should be very easy to modify to suit anyone's needs. The program is designed to be highly interactive and very easy to use.

Running EASY CHART

To run the program boot the back side of the AMPERCHART diskette and choose EASY CHART from the menu. Alternatively, type: RUN EASY CHART.

If you want to move EASY CHART to another disk, you must also move AMPERCHART.TB to the other disk. EASY CHART will not run without this file being present.

Entering and Editing Data

Once up and running, you will see a table of X and Y values, originally all set to zero. (If you have an Apple //e, you will notice that the program automatically switched you into 40 column mode. You may want to note the technique used to accomplish this switch as it is not a trivial task to do correctly in all cases.) You can enter data into the table by just typing.

Data is accepted into the table by moving the cursor to another cell. Think of it like a very narrow spreadsheet. The arrow keys, the RETURN keys, and all other unassigned keys will in general move the cursor. Use the RETURN key to advance to the next row. Use the space key to advance to the next column. Use arrow keys to move up and down the list. The list is limited to 50 rows and if you try to go past the end of the list the computer will beep at you.

There are several things you can do to the data in the list. These are indicated by the menu at the bottom screen. The notation (Z)ERO means that hitting the Z key will perform the ZERO operation, namely place zeros in the X array.

The commands at this point are:

(A)CCEPT - Accept all data in the rows from row 1 up to and including the row the cursor is on.

(D)ELETE - Delete the row the cursor is sitting on and move all the other rows up one notch. Zeros are entered into row 50.

(I)NSERT - Insert a row above where the cursor is sitting and move all the rows from the cursor to row 50 down one notch. Whatever was in row 50 is lost.

(Z)ERO - Zero the X column.

(S)WAP - Swap the X and Y columns.

(C)OUNT - Insert the numbers 1 through 50 into the X column, wiping out its previous contents.

(W)RITE - Write the data out to disk in a standard text file format.

(R)EAD - Read data into the list from a disk file.

Plotting the Data

Once you are happy with the data (and if you make a mistake you can always go back) hitting and acknowledging the (A)CCEPT command will lead you into the next section. You will then have to choose what type of plot you want BAR, PIE or LINE. Pick one.

Each of these choices will result in a plot of the desired type being made automatically on the screen. The computer in its infinite wisdom will choose a set of parameters to start with and will plot the data to the screen. It will then present you with a set of options to edit or redraw the picture, go back to the data editor, go to another plot type, or give up and leave the program. The choices are:

(L)ABEL - Add vertical, horizontal, or sideways labels to the chart.

(A)REA ERASE AND STUFF - Select a box on the screen and clear, fill, or reverse everything in that box.

(F)ILL - Fill any convex closed area on the screen in any color you want.

(C)LEAR - Clear the screen and redraw the basic chart using the current parameter list.

(M)ODIFY - Change the parameters that the computer used to draw the chart. This is the most powerful of the commands and offers you the most flexibility in chart formatting.

(E)DIT DATA - Return to the data editor.

(D)UMP - Dump your plot out to an Epson MX or FX series printer. This assumes that you have an Apple, Epson, or Tymac parallel interface in slot 1. If not, you'll have to re-configure using PRINTER CONFIGURE (side 1), and then and re-install the correct dump routine in the program EASY CHART on your work disk using Routine Machine.

(N)EW PLOT - Go back to try a different plot on the same data or a way to get back to the original computer drawn plot if you've really monkeyed around with the plot parameters.

(R)EAD - Read a picture file into memory from the disk.

(W)RITE - Write the current screen (minus the command table at the bottom) out to disk.

(Q)UIT - Quit the program. To reenter without losing your data, type : GOTO 200

Now that we have seen what the overall structure is, let's take a closer look at several of the commands.

Plotting Parameters

The (M)ODIFY command causes the chart to be erased, leaving any labels, filled areas, or overlayed charts that are on the screen. A list of parameters that can be changed is then displayed on the screen. There are some parameters (e.g. Clipping window) that are displayed for all three plotting modes. Others (e.g. Pie slice offset, Bar chart fill) are particular to only one plot type. A quick run down of the parameters is given below, but the best way to see what they do is to try them.

CLIPPING WINDOW - Defines where the chart is drawn on the screen. The right must be greater then the left value, the top must be greater then the bottom, and all of the parameters must reside on the screen (0<=X<=279, 0<=Y<=191). If you don't follow these rules the program will ignore your requested change.

SCALING WINDOW - Defines the scale to which the chart will be drawn. Here you can enter any values you want as long as the left doesn't equal the right and the top doesn't equal the bottom.

AXES - Defines the spacing of the tic marks on the axes and where the axes cross each other. Notice that the meaning of the tic mark parameters change slightly if you select a log mode. The bars in the bar chart are drawn relative to the axes, not the bottom of the graph so you can get some neat effects by placing the y-origin in between the top and bottom scale.

GRID - In the line plot you can select seperate parameters for a grid of dots. The dots can either be sparsely or fully spaced. Try both.

DATA SOURCE - This allows you to plot either the X or the Y values for the pie chart and either the X or Y or both in the bar chart. If you choose both for the bar chart, the X & Y bars will be plotted next to each other.

FILL - The bar chart allows you to select automatic filling of the bars. If you are plotting both the X and the Y data, then only the Y bars are filled.

SLICE OFFSET - Set to zero for standard pie chart, and set between 5 and 15 for exploded pie charts.

LINE TYPE - Selects points, characters, connecting lines or vertical lines for the line chart.

LINEAR/LOGARITHMIC SCALING - Selects linear or logarithmic scaling for the line chart. You cannot take the log of a negative number or zero and the program will inform you of that fact if you try.

Labeling

Labeling is performed interactively by choosing a format (horizontal, vertical, or sideways), moving the label to where you want it using the cursor keys (I,J,K,M or arrow keys), then either accepting or

rejecting the final label. The cursor movement is the only tricky part. When moving a label, the cursor keys move the label, the RETURN key is used to accept the label location and the alphanumeric keys are used to select the step size with which to move the label.

The routine starts out shifting the label 5 dots for each press of the cursor keys. If you want to go faster, push one of the number keys 6-9 to go really fast. To get finer movements push the 1-4 keys or ('0' for 1 dot per keystroke motion). Any time a cursor is used in this program, the number keys have a similar effect.

Area Erase

Rectangles can be selected and filled, erased, or the contents reversed (useful for highlighting) using the AREA ERASE command. Just select the lower left corner first, and then watch the rectangle expand as you select the upper right corner. If you make a mistake, the routine also allows you to exit without changing the rectangle. Remember that the number keys can be used to increase or decrease the step size used to move the corners of the box.

Program Modification and Technical Information

The program is modular and entirely unlocked. You are encouraged to tinker with, modify or just look at the code to see how such a package can be easily written using AMPERCHART. You may be interested to know that this program took a total of just 4 days to write and document!

On the technical side, the program uses RELOCATE I.TB and AMPERCHART JUMP FILE.TB. AMPERCHART is loaded into memory from $800 to $2000, below Hi-Res page 1 and where Applesoft programs normally reside. Two Routine Machine modules are used in the program: SCROLL 4X.TB from the Routine Machine & Screen library disk and DUMP.TB. If EASY CHART was being developed as a standalone project, many more Routine Machine modules would have certainly been used. The program itself is loaded into memory starting at $4000, just above Hi-Res page 1.

With all of the routines in memory at once, the program is fast, but is somewhat limited in features and error checking. In its present configuration there are approximately 3000 bytes available for code or data expansion. Of course additional memory could be made available to expand sections of the code by deleting unused sections of the code (e.g. statistics and curve fitting could be added by eliminating the pie chart sections) or by moving DOS to the language card of a 64K machine.

DIF TO EASY CHART File Converter Utility

A simple file converter utility called DIF->EASY CHART FILE CONVERTER is included on the Routine Machine & Chart (side 2) diskette. This program allows you to tranfer data from many spreadsheet programs to EASY CHART. It is written in Applesoft and is fully commented so that you can see the techniques used to accomplish the file conversion.

To transfer data from Visicalc (for example) to EASY CHART, you must save the data in the DIF format using the /S0 command. Then run DIF->EASY CHART FILE CONVERTER and follow the directions. This will create an EASY CHART compatible file.

ADDING PRINTOGRAPHER TO EASY CHART

If you have The Printographer graphics printing program from Roger Wagner Publishing, you can add the graphics printing routine from it to EASY CHART to allow it to print on virtually any combination of printer and interface card.

This procedure should only be done to a COPY, never the original Chart 'n Graph diskette.

1 UNLOCK EASY CHART

2 LOAD EASY CHART

3 ADD LINE 2 AS FOLLOWS:

2 PRINT CHR$(4);"MAXFILES1"

4 DELETE LINES 20 THROUGH 120 IN EASY CHART

5 LIST LINE 5790

IT NOW SAYS: 5790 CALL IR"DUMP",1,15

CHANGE IT TO: 5790 CALL IR"DUMP",HGR,I,H,1,"C"

6 (OPTIONAL) Some print graphics files are larger than others, depending on the printer and interface card for which they are configured. Therefore, in order to be able to add the PRINT GRAPHICS file to EASY CHART, it may be necessary to remove all REM'S in EASY CHART. If you need to do this, first print out the program listing with the REM'S still in it for future reference. Then replace all REM'S from line 190 on with a colon (:).

For example, it now says:

190 REM SET UP ENTRY DISPLAY

CHANGE IT TO: 190:

Repeat for all successive REM'S

7 BRUN WORKBENCH

8 SELECT ITEM #2 (REMOVE A COMMAND) AND REMOVE THE COMMAND CALLED "DUMP"

9 SELECT ITEM #1 (ADD A COMMAND) AND ADD THE COMMAND CALLED "PRINT GRAPHICS" FROM YOUR PRINTOGRAPHER DISK, AND NAME IT "DUMP"

10 SELECT ITEM #0 (EXIT) TO EXIT WORKBENCH

11 BRUN REMOVE WORKBENCH

12 SAVE EASY CHART

13 LOCK EASY CHART

That's it! EASY CHART should now print fine on your printer. If you should change your printer type, slot (Apple II/II+/IIe only), or interface card, you will have to re-configure the Printographer diskette, and replace the PRINT GRAPHICS Toolbox file again.

APPENDIX E

SHAPE MAKER

Applesoft provides the capability for displaying and manipulating sets of shapes on the hi-res screen. Once a table of shapes has been defined, the user can load those shapes and display them using the DRAW, XDRAW, ROT, and SCALE commands. AMPERCHART also supports the use of shapes with its label commands and AMPERCHART Shape files.

The difficulty with shapes is not in the using but in the making. The SHAPE MAKER program has been included to alleviate this problem. It makes creating a shape table easy and quick. The program also allows you to add to or delete from existing shape tables, to view any shape table and to rearrange shapes within a table.

Individual shapes within a table are defined in terms of the path required to draw that shape. Most shape programs allow you to define an arrangement of pixels that will make up the shape and then they scan through those pixels automatically to derive the shape path for the table. This approach creates shape tables that are much larger (sometimes as much as twice the size) as is necessary. Other, less sophisticated shape programs, allow you to create the path directly. Unfortunately these programs are usually difficult to use. SHAPE MAKER combines the best of these two approaches to make it easy to create compact and efficient shape tables.

There are three steps involved in creating a shape using SHAPE MAKER. First, the arrangement of pixels in the shape must be defined. Secondly, the user must define a path that passes through all of the defined points. Finally the program analyzes the shape path and adds it to the current shape table in memory. The whole process takes only a few minutes for most shapes.

Once a shape table has been created and saved to disk, it can be used directly following the procedures described in the Applesoft manual or it can be used by AMPERCHART by converting it to an AMPERCHART Shape Table as described in Chapter 5 of this manual. Created shape tables can also by used by Shape Gobbler on the Wizard's Toolbox diskette.

Using SHAPE MAKER

To run SHAPE MAKER just boot on side 2 of the Chart 'n Graph Toolbox diskette and then select RUN SHAPE MAKER from the menu. Shape Maker can also be directly run from the immediate mode.

You must then choose between using a joystick or the keyboard. The keyboard gives more precise control, but both are easy to use.

Next you must give the maximum width and height of the shapes to be defined. The program can handle shapes up to 17 wide and 15 high. For example, the standard AMPERCHART character set is defined on a 5 wide by 7 high grid (and was designed using an early version of this program!)

Next you must choose the beginning coordinate of each shape in the table. Since shapes are nothing more than a path to follow for the machine, it makes a difference where you start to draw your shape. Most shapes are created in tables where you want the shapes to be drawn in a consistent manner relative to the cursor. In AMPERCHART, for example, all of the shapes are drawn from the bottom left corner of the grid.

If you are using a joystick, you choose the beginning coordinate to be used by moving the flashing cursor to the desired location and pressing either game button. With the keyboard, you can use the I,J,K,M keys to move the cursor to the desired location, then hit RETURN.

This concludes the initialization steps from the program. The main menu appears next. From this menu you can choose to either add a new shape, delete a shape, initialize (clear) the current table, view shapes in the current table, move a shape from one location to another within the table, load or save a table, or quit the program. To select a function just hit the indicated key.

Assorted Menu Functions

The initialize and quit commands ask for verification before proceeding to guard against accidents.

The delete shape and move shape commands both use the number of a particular shape to identify it. Thus

you'll need to determine the number of the shape you want to delete or move and, in the case of the move, the new position for the shape to be inserted into. This can be done using the view command.

The view command allows you to see all of the shapes currently defined as well as their position in the shape table. If there are no shapes in the current table, this command will not function.

To load or save a shape, just follow the directions in the load or save commands. If you load a shape file, the old current file is written over and lost.

Adding a Shape

Shapes can be added to the current shape table using the add command. The first step, after hitting 'A' from the main menu, is to define the pixels that are on in the shape. To do this just move the cursor around on the displayed grid (using the joystick or the I,J,K,M keys as appropriate) and turn on or off the desired pixels (using game buttons 0 or 1 on the joystick or the 'D' & 'E' keys). The way the shape will look is simultaneously displayed on the right side of the screen. If you want to abandon this attempt at a shape, hit 'X' to exit. Once you are happy with the way the shape looks hit 'F' to indicate you're finished.

After hitting 'F', a grid of large size pixels will be displayed on the screen corresponding to the shape that you just drew. You must now select a path that connects all of the points that you defined. The path that you select must go through all of the points (the program doesn't check to see if you missed any).

If you are using the keyboard option, simply use the I,J,K,M keys to move the cursor around the grid. The selected path is designated by '+' markers between the pixels in the grid. To erase a portion of the path just move backwards retracing your path.

Use of the joystick, although faster, is a bit more complicated. First you must move the joystick over to the highlighted pixel that is the beginning coordinate for the shape. After you have done this, the joystick will change modes and allow you to move only one step at a time to either side of the current position.

After moving the cursor one step, you can set the path by hitting game button 1. To erase a section of path, carefully retrace your steps, at each point hitting game button 0.

After the path is completely defined, hit 'F' to indicate you are finished. The program will then process the shape and add it to the table. If you make a mistake you can always delete it and try again.

Several points should be noted about defining paths. First, if you define a path that traces over itself, it is very easy to get lost. You must then use the path length indicator at the bottom of the screen and try to erase your way out of the overlapping sections. Secondly, paths that trace over the same point more than once can cause problems when using the XDRAW command. Its best to avoid overlapping paths of all kinds. Third, a point is drawn when the path leaves that point for another.

Thus, if you don't force the path to go past the last point, then the last point will not appear. Finally, for technical reasons, paths that move straight up while passing through blank pixels are not particularly efficient in terms of storage. The program handles these cases since there are times that they cannot be avoided, but in general its best to avoid blank upward moving paths.

Joystick Files

There are two ways in which joysticks can be configured. SHAPE MAKER comes set up for a standard joystick (meaning the kind that the author happens to own). If you find the controls on your joystick operate strangely no matter which way you orient the joystick, then typing EXEC ALTERNATE JOYSTICK, from the immediate mode (not within SHAPE MAKER or another program) and with side 2 of the disk in the drive, will rewrite the sections of SHAPE MAKER necessary to adapt SHAPE MAKER for your joystick. Another EXEC file, STANDARD JOYSTICK is also included to undo the changes made by ALTERNATE JOYSTICK.

Programming Notes

The SHAPE MAKER is run using RELOCATE II.TB so that both Hi-Res pages are protected. The display that you see resides on Page 1. The current shape table is

stored in the area used for Page 2. Thus the shape table length is limited to just over 8000 bytes.

The program is written entirely in BASIC and, although its complex, it is commented and modifiable. Be aware though, that two special Toolbox commands are attached to the program to speed up its shape table handling so the program should not be renumbered. (The renumbering process strips off the Toolbox commands.) Futhermore, the ALTERNATE JOYSTICK and STANDARD JOYSTICK files rewrite sections of the program so if you do modify the program don't disturb lines 440, 1050, 1590, and 1785.

The program requires the use of AMPERCHART.TB on the back of the AMPERCHART diskette. If you move SHAPE MAKER to another diskette, you must move this file as well.

AMPERCHART COMMANDS	PG REF.
ARC(radius,angle,sides,number)	29,52
AXS(xtic,ytic,xorigin,yorigin)	23,47
BIT(sx,sy,colorv)	32,46
CLP(sxmin,sxmax,symin,symax)	20,40
CLR	22,43
DRW(ux,uy)	17,45
DSP(page)	29,39
FCL	22,43
FIL(ux,uy)	31,51,172
FRM	22,53
GID(xtic,ytic,xorigin,yorigin)	25,48
GGO(page)	14,36
GMD	15,37
HLB(string$,ux,uy,position)	26,49
LMD(mode)	32,42
LOC(address)	33,54
MIX	15,38
MOV(ux,uy)	17,44
NMX	15,38
PLT(ux,uy)	17,45
RVS	16,44,172
SCL(uxmin,uxmax,uymin,uymax)	18,41
STU(sx,sy,uxv,uyv)	32,54
TMD	15,37
TSZ(xsize,ysize)	25,53
VLB(string$,ux,uy,position)	26,50
WRK(page)	29,39

NOTATION

sx,sy - Screen coordinates (0-269,0-181) (0,0) is in lower left corner. 16,18,54

ux,uy - User coordinates defined by the scale command (SCL). 18,54

CALLING SYNTAX

Workbench Installed Option: 7,14,102

```
&"NAME",COMMAND(VARIABLE LIST)
```

Stand Alone Option: 55,105

```
&COMMAND(VARIABLE LIST)
```

GRAPHICS MODE COMMANDS — PG REF.

GGO(page) — 14,36
Graphics Go - Initialize graphics.

TMD — 15,37
Text Mode - Set display to text mode.

GMD — 15,37
Graphics Mode - Set display to hi-resolution graphics mode.

MIX — 15,38
Mixed Mode - Set display to mixed hi-resolution and text.

NMX — 15,38
No Mix - Reset display to hi-resolution only.

DSP(page) — 29,39
Display - Select page to be displayed.

WRK(page) — 29,39
Work - Select page to be drawn on.

CLIP & SCALE COMMANDS — PG REF.

CLP(sxmin,sxmax,symin,symax) — 20,40
Clip - Set clipping window.

SCL(uxmin,uxmax,uymin,uymax) — 18,41
Scale - Set scaling for current clipping window.

LMD(mode) — 32,42
Log Mode - Set log mode.

mode	x-axis	y-axis
0	lin	lin
1	log	lin
2	lin	log
3	log	log

CLEAR & REVERSE COMMANDS	PG REF.

CLR — 22,43
Clear - Clear current clipping window.

FCL — 22,43
Full Clear - Clear entire screen.

RVS — 16,44,172
Reverse - Reverse current clipping window.

BASIC PLOTTING & BIT READ COMMANDS	PG REF.

MOV(ux,uy) — 17,44
Move - Move to point ux,uy.

PLT(ux,uy) — 17,45
Plot - Plot point at ux,uy.

DRW(ux,uy) — 17,45
Draw - Draw line from current cursor position to ux,uy.

BIT(sx,sy,colorv) — 32,46
Bit - Read screen bit at sx,sy and place value in colorv.

colorv	color
0	black1
1	green
2	blue
4	black2
5	orange
6	blue

AXES & GRID COMMANDS (cont.)	PG REF.

AXS(xtic,ytic,xorigin,yorigin) — 23,47
Axes - Draw an X-Y axis in clipping window. Axes intersect at point xorigin,yorigin. In linear mode, xtic or ytic specify distance between tic marks. In log mode, xtic or ytic specify the number of tic marks per decade (xtic or ytic=1,2,5, or 10 yield 10,5,2, or 1 tic marks per decade respectively).

AXES & GRID COMMANDS (cont.) — PG REF.

GID(xtic,ytic,xorigin,yorigin) — 25,48

Grid - Draw a grid in clipping window. Parameters are defined the same as in the AXS command.

LABELING COMMANDS — PG REF.

HLB(string$,ux,uy,position) — 26,49

Horizontal label - Write the label string$ horizontally at point ux,uy. The relative position of ux,uy and the label is determined by position.

VLB(string$,ux,uy,position) — 26,50

Vertical label - Write the label string$ vertically at point ux,uy. The relative position of ux,uy and the label is determined by position.

position	location of string$ relative to ux,uy
1	left & below
2	left centered
3	left & above
4	centered & below
5	centered
6	centered & above
7	right & below
8	right centered
9	right & above

MISCELLANEOUS COMMANDS — PG REF.

FIL(ux,uy) — 31,51,172

Fill - Fill a black convex area about point ux,uy.

ARC(radius,angle,sides,number) — 29,52

Arc draw - Draw a polygon with the specified radius and total number of sides. The starting angle of the arc is specified in radians. Of the total sides in the polygon, only number are drawn.

MISCELLANEOUS COMMANDS	PG REF.
TSZ(xsize,ysize) Tic Size - Specify tic size for axes command in screen coordinates.	25,53
FRM Frame - Draw a frame around the current clipping window.	22,53
STU(sx,sy,uxv,uyv) Screen-to-user coordinates - Convert screen coordinate sx,sy to user coordinates and return in variables uxv,uyv.	32,54
LOC(address) Location - Return current address of Amperchart.	33,54

OTHER COMMANDS	PG REF.
RELOCATE.I.TB and RELOCATE II.TB &"RELOCATE"	12,57,100,102
AMPERCHART JUMP FILE.TB &"CHART",Command [,Variable list]	12,57,102
SPLITTER.TB &"SPLIT",splitflag	59,104

splitflag	effect
0	unsplit
1	split around page 1
2	split around page 2
3	split around both pages

BLOAD PIC.TB &"BLOAD",filename$,pagenumber	63
BSAVE PIC.TB &"BSAVE",filename$,pagenumber	64
WINDOW.TB &"WINDOW",dir,xbgn,xend,ymin,ymax,page,intarray	65

dir	effect
0	array -> graph
1	graph -> array with clear
2	graph -> array without clear

HI/LO ZOOM.TB 68

&"ZOOM",x,y,zoomflag {,size)

zoomflag	effect	size	box
0	unzoom	0	40x48
3	white1 zoom	1	40x40
7	white2 zoom		

HI-RES EXPAND.TB 70

&"EXPAND",sxmin,sxmax,symin,symax

MEDIAN FILTER.TB 71

&"FILTER",sxmin,sxmax,symin,symax,threshold

threshold = (0-9)

THREE-D ARRAY.TB 73

&"THREED",A,B,G,SX,SY,SZ,XT,YT,ZT,ZO

DUMP.*.TB 76

&"DUMP"{,pagenumber}{,tabposition}

filename	pagenumber	tabposition	
		pn<3	pn=3
MX80.S1P1	0-2	0-33	NA
MX80.S1P2	NA	NA	NA
MX.S3P1	0-2	NA	NA
MX100.S1P12	0-3	0-90	0-43
9500.S1P12	0-3	0-85	0-38
9500.S2P1	0-2	0-26	NA
9501.S1P12	0-3	0-94	0-57
9501.S2P1	0-2	0-57	NA
IMAGE.S1P12	0-3	0-41	0-1

TONE.TB 6,82

&"TONE" {,pitch} {,duration] {;pitch2,durtn2}

&"TONE"

&"TONE",P,D

&"TONE",P,D;P2,D2

RESTORE AMPERSAND.TB 84,164

&"AMP"

UC & LC SHAPE.TB 90

S10X14 SHAPE.TB 91

APPLE ICONS.TB 93

&"AMPERCHART",LOC(ADR) : &"NEW SHAPES",ADR

OTHER COMMANDS (Cont.) PG REF.

IDENTIFICATION.TB 80
&"ID",result {result = R1 + R2}

R1	configuration
128	//e with extended 80 column card
64	//e with standard 80 column card
32	//e but no Apple 80 column card
0	Not an Apple //e

R2	type
0	//e without Mouse Icon support (Old ROMs)
1	//e with Mouse Icon support (New ROMs)
2	//c
3	Unknown member of // family.

DOUBLE HI-RES COMMANDS PG REF.

DHR MODIFIER.TB 113,138
&"MODIFY"

DHR ROUTINES.TB 113,132
DHR ROUTINES 113,122
&"DHR",HGR
&"DHR",HPLOT X,Y TO X1,Y1
&"DHR",DRAW SH AT X,Y
&"DHR",XDRAW SH AT X,Y
&"DHR",HCOLOR=15
&"DHR",LNTYP=3

DHR RELOCATE.TB 115,120
&"RELOCATE"

DHR RELOCATE.MOD.TB 113,137
&"RELOCATE"

DHR JUMP FILE.TB 113,137
(see DHR AMPERCHART)

DHR AMPERCHART 113,116,121
DHR AMPERCHART.MOD.TB 117,145
DHR AMPERCHART.MOD 113,143
&"AMPERCHART",Command [,Variable list]

DHR DISPLAY.TB 124
&"DISPLAY",page

DHR LOAD.TB 125
&"DLOAD","filename",pagenum

DOUBLE HI-RES COMMANDS — PG REF.

DHR SAVE.TB — 126

```
&"DSAVE","filename",pagenum
```

DHR WINDOW.TB — 127

```
&"WINDOW",dir,xbgn,xend,ymin,ymax,1,intarray
```

dir	effect
0	array -> graph
1	graph -> array with clear
2	graph -> array without clear

DHR TRANSFER.TB — 130

```
&"TRANSFER",source,destination,dir
```

dir	effect	source, destination = (1-5)
0	card -> main	
1	main -> card	

DHR PRINTER DUMPS — 134

```
&"PRINT",1,tab position,[mode]
```

AMPERCHART INSTALLATION

WORKBENCH - STANDARD OPTION — 12,57,102

Follow these steps:

1) BRUN WORKBENCH and add
   AMPERCHART JUMP FILE.TB and
   RELOCATE I.TB or RELOCATE II.TB
2) BRUN REMOVE WORKBENCH
3) Make sure AMPERCHART.TB is on the disk.
4) RUN your program:

```
 1 CALL PEEK(175) + 256 * PEEK(176) - 46
 2 &"RELOCATE"
10 &"CHART",GGO(1)   {or GGO(2)}
```

WORKBENCH - SPLITTER OPTION — 59,104

Add AMPERCHART.TB like any other Toolbox command. If SPLITTER.TB is not used, then you can only use page 2 of Hi-Res. If you add SPLITTER.TB, then both Hi-Res pages can be used.

AMPERCHART INSTALLATION — PG REF.

AMPERCHART ONLY: (Page 2 only) — 104

```
 1 CALL PEEK(175) + 256 * PEEK(176) - 46
10 &"CHART",GGO(2)
```

AMPERCHART+SPLITTER (Page 1 or 2) — 104

```
 1 CALL PEEK(175) + 256 * PEEK(176) - 46
 2 &"SPLIT",1  (or 2)
10 &"CHART",GGO(1)   {or GGO(2)}
```

STAND ALONE OPTION - HIGH MEMORY — 55,105

```
 1 HI=PEEK(115)+256*PEEK(116)-6131
 2 PRINT CHR$(4);"BLOAD AMPERCHART.TB,A";HI
 3 AH=INT(HI/256): AL=HI-AH*256
 4 POKE 1014,AL: POKE 1015,AH
 5 HIMEM: HI
```

STAND ALONE OPTION - LOW MEMORY — 107

From outside a program:

```
Either BRUN RELOCATE I.TB
or     BRUN RELOCATE II.TB
```

To remove, type: FP

From within a program:

```
 1 PRINT CHR$(4);"BRUN RELOCATE I.TB"  {or II.TB}
 2 AD=2048
10 CALL (AD)GGO(1)       {or GGO(2)}
99 PRINT CHR$(4);"FP"
```

DHR AMPERCHART INSTALLATION

On an Apple //c or //e, follow these steps: 112,117,137

1) BRUN WORKBENCH and add
   DHR JUMP FILE.TB and DHR RELOCATE.TB.
2) BRUN REMOVE WORKBENCH
3) Make sure DHR AMPERCHART is on the disk.
4) RUN your program:

```
 1 CALL PEEK(175) + 256 * PEEK(176) - 46
 2 &"RELOCATE"
10 &"CHART",GGO       {DHR Hi-Res page 1 only}
20 &"CHART",HCOLOR=15
```

DHR AMPERCHART INSTALLATION

PG REF.

On an Apple //e only, follow these steps: 112,117,137

1) BRUN WORKBENCH and add
   DHR JUMP FILE.TB and
   DHR RELOCATE.MOD.TB
2) BRUN REMOVE WORKBENCH
3) Make sure DHR AMPERCHART.MOD and
   DHR MODIFIER.TB is on the disk.
4) RUN your program:

```
 1 CALL PEEK(175) + 256 * PEEK(176) - 46
 2 &"RELOCATE"
10 &"CHART",GGO        {DHR Hi-Res page 1 only}
20 HCOLOR=15
```

CHART 'N GRAPH TOOLBOX™

By Andy Scheck & Rick Chapman

Chart 'N Graph Toolbox makes putting charts and graphics in your own programs **quick and easy**. No need to master Hi-Res graphics, with Chart 'N Graph you can transform your data into custom charts, graphs, or plots, and make them appear the way **you** want.

With 40 special graphics commands, Chart 'N Graph is faster and easier to use than Applesoft. No mysterious PEEKS, POKES or CALLS, just **easy-to-understand commands**. For example, clearing the Hi-Res screen is as simple as:

```
100 & "CHART", CLR
```

and the Hi-Res screen clears for you. Suppose you want to draw a frame around your chart. Just add the following command to your program:

```
110 & "CHART", FRM
```

and instantly a frame appears around the Hi-Res picture you were creating. Other **handy commands** include Automatic Scaling, Axes Generation, Area Fill, Move, Plot, Draw, Window Frame, Window Clear, Clipping Window, Window Reverse, Tic Marks, Grid Pattern, Zoom/Unzoom, Horizontal Labels, Vertical Labels, and more!

Chart 'N Graph also contains a variety of additional graphics support commands. These include special **Double Hi-Resolution** commands for the Apple IIc or an Apple IIe with an Extended Memory 80 column card. These Double Hi-Res commands produce accurate and attractive charts, including bar or pie charts in **sixteen colors**! And these Double Hi-Res commands are just as simple and easy to use as the others.

With Chart 'N Graph you can print your charts on an Epson or Imagewriter printer, create windows, zoom from Hi-Res to Lo-Res and back, and even do 3-D plotting. There is also a command to automatically split your program around the Hi-Res pages.

As a **special bonus**, the package includes **E Z CHART**, a free, complete charting program that converts your data into handsome bar, pie or line charts. And since it is unlocked, you can even list the program and modify it to meet your specific needs.

Finally, for your convenience, Chart 'N Graph is unlocked, copyble and hard disk compatible. The Chart 'N Graph Toolbox is your key to custom chart graphics on your Apple.

SYSTEM REQUIREMENTS:
48K Apple II, II+, IIe or IIc

Ask your dealer for other Toolbox packages including Database Toolbox, Video Toolbox, and Wizard's Toolbox.

Roger Wagner™ PUBLISHING, INC.

ISBN 0-927796-20-1